4

Massachusetts
Institute of
Technology

Laboratory for Computer Science Progress Report

26

July 1988-
June 1989

The work reported herein was carried out within the Laboratory for Computer Science, an MIT interdepartmental research laboratory. During 1988-89, our principal financial support has come from the Defense Advanced Research Projects Agency (DARPA). DARPA has been instrumental in supporting most of our research during the past 26 years and is gratefully acknowledged here. Our overall support has come from the following organizations:

- Defense Advanced Research Projects Agency;
- National Institutes of Health;
- National Science Foundation;
- Office of Naval Research; and
- MIT controlled IBM funds under the IBM/MIT joint study contract.

Other support of a generally smaller level has come from Apple, AT&T, Nynex, Siemens, Texas Instruments, United States Army Research Office, Xerox, and the multiple industrial members of the MIT X Consortium.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) MIT/LCS/PR - 26	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION MIT Lab for Computer Science / 6b. OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research/Dept. of Navy
6c. ADDRESS *(City, State, and ZIP Code)* 545 Technology Square Cambridge, MA 02139	7b. ADDRESS *(City, State, and ZIP Code)* Information Systems Program Arlington, VA 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION DARPA/DOD / 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS *(City, State, and ZIP Code)* 1400 Wilson Blvd. Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO

11. TITLE *(Include Security Classification)*
MIT Laboratory for Computer Science Progress Report - 26

12. PERSONAL AUTHOR(S)
M.L. Dertouzos

13a. TYPE OF REPORT Technical /Final	13b. TIME COVERED FROM 7/88 TO 6/89	14. DATE OF REPORT *(Year, Month, Day)* June 1989	15. PAGE COUNT 309

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Artificial Intelligence, Computer Architecture, Computer Science, Electrical Engineering, Network, Theory Computers, Programming Languages

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Annual Report of Progress and Final Technical Report Under Contracts:

a.) N00014-83k - 0125, Darpa Order 5602/2095
b.) N00014-84k - 0099, Darpa Order 4920

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Carol Nicolora	22b. TELEPHONE *(Include Area Code)* (617) 253-5894	22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

*U.S. Government Printing Office: 1985--507-047

Unclassified

Massachusetts Institute of Technology
Laboratory for Computer Science
545 Technology Square
Cambridge, Massachusetts 02139
617-253-5851

Progress Report

26

Contents

Introduction

ADMINISTRATION

Academic Staff

M. Dertouzos, Director
R. Rivest, Associate Director
A. Vezza, Associate Director

Administrative Staff

G. Brown, Facilities Officer
A. Djanzni, Sr. Staff Accountant
W. Fitzgerald, Fiscal Officer
M. Mitchell, Administrative Officer
D. Ruble, Staff Accountant
M. Sensale, Librarian
P. Vancini, Personnel Officer

Support Staff

J. Canaya	P. Mickevich
L. Cavallaro	R. Purvis
C. Conkey	M. Rebelo
S. Costanza	C. Robinson
R. Donahue	M. Siddique
C. Ehling	D. Santoro
C. Houser	J. Vermette
H. Kruh	L. Wenger
J. Little	

1.1 Overview

The MIT Laboratory for Computer Science (LCS) is an interdepartmental laboratory whose principal goal is research in computer science and engineering.

In 1963, when the Laboratory was founded as Project MAC, it explored and developed one of the world's earliest timeshared computer systems. This 1960's research on the Compatible Time Sharing System (CTSS), and its successor MULTICS, contributed innovations like the writing of operating systems in high level programming languages, virtual memory, tree directories, online scheduling algorithms, line and page editors, secure operating systems, concepts and techniques for access control, computer aided design, and two of the earliest computer games—space wars and computer chess.

These early developments laid the foundations for the Laboratory's work in the 1970's on knowledge based systems, for example, the MACSYMA program for symbolic mathematics, on natural language understanding; and in the development (with BBN) and use of packet networks. In the 1970's, the Laboratory also developed theoretical results in complexity theory and linked cryptography to computer science through concepts and algorithms for public encryption (RSA). In the late 1970's, Project MAC, renamed as the Laboratory for Computer Science (LCS), embarked on research in such areas as clinical decision making, in the exploration of cellular automata at the borderline between physics and computation, and on the social impact of computers. At the same time, the Laboratory began two major research programs in distributed systems and languages, and in parallel systems. These led to the notions of data abstractions and the Clu language; the Argus distributed system; the dataflow principle and associated languages, and architectures of parallel systems; local area ring networks; to program specification; and workstation development, where the Laboratory contributed the earliest UNIX ports and compilers and the NuBus architecture, now used in commercial computers like Apple's Macintosh II.

The Laboratory's current research falls into four principal categories, Parallel Systems; Systems, Languages, and Networks; Intelligent Systems; and Theory. The principal technical goals and expected consequences in each of these four categories are as follows:

In *Parallel Systems*, we strive to harness the power and economy of numerous processors working on the same task. Research in the area involves the analysis and construction of various hardware architectures and programming languages that yield, over a broad set of applications, cost-performance improvements of several orders of magnitude relative to single processors. This research is expected to affect most of tomorrow's machines which we expect to be of the multiprocessor variety—not only because of potential cost performance benefits but also because of the natural, yet unexploited, concurrence that characterizes contemporary and prospective applications from business to sensory computing.

In *Systems, Languages, and Networks*, our objective is to provide the concepts, methods, and environments that will enable heterogeneous computers, each working on different tasks, to communicate efficiently, conveniently, and reliably with one another in order to exchange information needed and supplied by their respective programs. Such communication may involve, beyond conventional electronic mail and file transfer, the calling of programs in one environment from programs in another, perhaps different, environment and the sharing of structured data among such programs. This research is also expected to have a broad impact on future systems, since virtually every machine will be connected to a network.

Taken together, these two thrusts in parallel and networked machines signal our expectation that future computer systems will consist of multiprocessors interconnected by local and long haul networks, and perhaps someday by national network infrastructures as ubiquitous and as important as today's telephone and highway infrastructures.

In the *Intelligent Systems* area, our technical goals are to understand and construct programs and machines that have greater and more useful sensory and cognitive capabilities. Examples include the understanding of spoken messages, systems that can learn from practice rather than by being explicitly programmed, and programs that reason about clinical issues and help in clinical decision making. We expect tomorrow's intelligent systems to be easier to use than today's programs across a broad front of applications.

In our fourth category of research, *Theory*, we strive to understand and discover the fundamental forces, rules, and limits of computer science. Theoretical work permeates many of our research efforts in the other three areas, for example, in the pursuit of parallel algorithms and in the study of fundamental properties of idealized parallel architectures and computer networks. Theory also touches on several predominantly abstract areas, like the logic of programs, the inherent complexity of computations, and the use of cryptography and randomness to the formal characterization of knowledge. The impact of theoretical computer science upon our world is expected to continue its past record of improving our understanding and helping us pursue new frontiers with new models, concepts, methods, and algorithms.

1.2 Highlights of the Year

The year 1988 marked the 25th Anniversary of our Laboratory. The occasion was celebrated with a two day symposium on current research for an international audience of some 1000 people, and a testimonial banquet attended by over 1200 members and guests of both the LCS and AI laboratories. Chaired by Professor Albert R. Meyer, the celebration was memorable and successful.

Research highlights during the reporting period were as follows:

1. Dr. Victor Zue and his research group moved from MIT's Research Laboratory of Electronics to LCS. This move promises to be significant for both the speech research effort and for other LCS groups through the potential synergism between speech research and computer architecture.

2. We concluded a major agreement with Motorola to build the Dataflow Machine, conceived and designed by the Computation Structures Group. This effort is significant for it represents the first major test of this new architecture invented by our Laboratory.

3. A new group, the Computer Architecture Group, was formed. It includes Professors Agarwal, Dally, and Ward, their students and staff. This large architectural group of computer architects plans to embark on a new project, NuMesh. The NuMesh involves an interconnection and intercommunication standard that goes beyond the notion of a computer bus to three dimensional structures. As currently envisioned, the NuMesh will consist of small cubes (about 2cm on the side) which will be able to plug together with other similar cubes at all six of their sides. Each cube will contain processing and communication chips. We envision that users of this technology will first construct in Tinkertoy fashion a special purpose aggregate that is best suited to their problems, and will then run these problems on the so-constructed machine. This approach is thus expected to make possible the benefits of special purpose computation out of general purpose subsystems. We are planning to fund this research out of an industrial consortium of manufacturers.

During 1988-89, the Laboratory continued its successful Distinguished Lecturer Series with presentations by David L. Parnas, Professor of Computing and Information Science, Queen's University; David S. Johnson, Department Head, Mathematical Foundations of Computing, AT&T Bell Laboratories; Raj Reddy, Director of Robotics Institute, Carnegie Mellon University; and Robert W. Taylor, Director, Systems Research Center, Digital Equipment Corporation.

During this reporting period, Professor David L. Tennenhouse joined the Advanced Network Architecture Group; Drs. Gregory Papadopoulos and Gill Pratt became Research Associates in the Computation Structures and the newly formed Computer Architectures Group, respectively. Dr. Victor Zue and his Spoken Language Systems Group, including four research scientists, 15 students, two visitors, and two support staff also joined the Laboratory. Two staff accountants, Ms. Azi Djazani and Mr. David Ruble joined the LCS administrative staff and Ms. Mary Mitchell joined MIT and LCS Administrative Officer of the Laboratory.

The Laboratory is organized into 18 research groups, an administrative unit, and a computer service support unit. The Laboratory's membership includes a total of 350 people—105 faculty and research staff, 35 visitors, affiliates, and postdoctoral associates, 30 support staff, 125 graduate students, and 55 undergraduate students. The academic affiliation of most of the Laboratory's faculty and students is with the Department of Electrical Engineering and Computer Science (EECS). The funding is predominantly from the U.S. Government's Defense Advanced Research Projects Agency, which accounts for about half of the total. The Laboratory is also funded by and has extensive links with industrial organizations. These include partnerships for the construction of major hardware systems, consortia for the development and maintenance of standards, like X Windows, and joint studies on research areas of common concern. Technical results of our research in 1988-89 were disseminated through publications in the technical literature, through Technical Report numbers 426 through 453, and Technical Memoranda numbers 363 through 400.

Advanced Network Architecture

Academic Staff

D. Clark, Group Leader	D. Tennenhouse

Research Staff

J. Davin	K. Sollins

Graduate Students

D. Feldmeier	R. Perlman
E. Hashem	T. Shepard
A. Heybey	L. Zhang

Undergraduate Student

D. Martin

Support Staff

A. Aldrich

Visitor

H. Rebstock

2.1 Introduction

The Advanced Network Architecture project continues to explore a number of problems related to the design of advanced data networks. As networks get bigger and faster, it is important to explore new design approaches, since the current assumptions and protocols may not scale well to match the expectations of tomorrow.

The central problem of our group has been the management of resources within the network: bandwidth, switching capacity, and buffering. If we are to achieve higher speeds and larger size, the tradeoffs among these resources must change, and new algorithms and approaches will be needed.

In the following sections, a number of specific projects related to this overall goal are described.

2.2 Network Control Algorithms

Lixia Zhang has nearly completed work on a new network architecture which integrates resource management and traffic control into the system. The new architecture can support a wide variety of applications and incorporate new technologies. It includes three basic parts: an elemental data transmission entity called a *flow*, an interface between users and the network which allows a flow to specify a set of performance attributes, and a distributed control algorithm that regulates network traffic to ensure overall performance.

2.3 Rate-based Flow Control

Previously, several members of the group designed a new transport protocol, NETBLT, which contained novel algorithms for flow control and error recovery [79]. In particular, the protocol contained a flow control algorithm based on rate regulation, rather than window permissions. Rate control is expected to provide smoother and more effective utilization of high bandwidth, long delay links.

The first version of that protocol did not have an algorithm for dynamic adjustment of the rate, but instead used manual adjustment. While manual adjustment was sufficient for a first set of experiments, it was not the basis for a practical system. Subsequently, Mark Lambert proposed a number of dynamic rate adjustment algorithms.

Helmut Rebstock, a visiting scientist from Siemens Corporation, assisted by James Davin, used the interactive network simulator to explore the behavior of these proposed algorithms for dynamic adjustment of transmission rates in NETBLT. Rebstock began a study of the capacity of adaptive variants of NETBLT to support equitable sharing of bandwidth among connections in a congested network.

Andrew Heybey performed experiments to extend the results reported previously in his undergraduate thesis [161]. A form of slow start was added to Mosely's rate-based protocol [239] and was shown (through simulation) to increase the percentage of the link capacity that can be used. The most important result is the observation that the protocol (with or without slow start) operates in either a stable or unstable region. As the fraction of the link's capacity that the protocol attempts to use is increased, the queue lengths abruptly change from an average of approximately five to wild oscillation between zero and several thousand. (In these experiments, there is no limit on queue length, and no packets are ever dropped).

2.4 Fair Queueing in Gateways

James Davin and Andrew Heybey experimented with a novel "Fair Share" queueing algorithm developed at Xerox PARC [94]. They simulated the described algorithm and verified its correct operation in several simple network topologies both with and without the presence of ill behaved users. In its simplest form, the algorithm enforces fairness—no user may use more than its fair share of the output bandwidth. It can also be used to enforce policy by giving some users a larger share of the bandwidth than others. Because the present algorithm only comes into play when the output queue length in the switch is greater than zero, the extension of fair queueing for operation on an underutilized link is being contemplated. Packet discard strategies are also being studied.

2.5 Protocol Performance Studies

David Clark engaged in a study of TCP processing overhead that strongly suggests that the details of TCP are not a central issue in host level processing. The results of this study indicate that, with current RISC processors, the specific TCP processing steps would permit packet transmission at a large fraction of a gigabit per second. This work, performed jointly with V. Jacobson at LBL, H. Salwen at Proteon, Inc., and J. Romkey, is reported in [82].

Eman Hashem studied packet clustering in network environments similar to those of the Internet. This work aimed at analyzing the causes and consequences of packet aggregation and its effect on congestion. Two major causes were identified: the TCP slow start retransmission strategy, and the interaction of the gateway and TCP congestion control mechanisms.

The slow start algorithm opens the TCP sender window exponentially by incrementing the window size by one for each acknowledgment received. This behavior leads to the clustering of the packets belonging to each TCP connection that is newly opened or is recovering from congestion. This phenomenon is not very harmful on its own, as the clustering persists only as long as the window is being opened. In a heavily loaded network, however, the TCP slow start algorithm and gateway congestion control schemes interact so as to increase the frequency of exponential window sizing. The gateway signals congestion by discarding packets in excess of its capacity. TCP responds to this signal by shutting its window and reopening it exponentially. Following recovery from packet loss, the TCP continues to open the window linearly. Eventually, the previous level of congestion is again realized, and the recovery process is repeated. Thus, the network oscillates between congestion recovery and load optimization, causing packets from most connections to aggregate at the bottleneck resources during each congestion cycle. This global effect, involving all connections, coupled with local clustering of packets from the same connection, leads to a performance degradation. Long end-to-end delays result from the high queueing delays incurred at the bottleneck resources, and throughput decreases owing to bandwidth wasted in retransmissions. The extent of the packet clustering effects depends on the details of the congestion schemes and on how quickly they react to congestion. Although slow start encourages packet clustering, it minimizes wasted bandwidth by dynamically adjusting its window size to the current network load limit.

Another approach to studying TCP behavior was pursued by Timothy Shepard. A system for collecting and storing about 12 hours of the protocol headers of all the packets on one of the main Ethernets in the Laboratory has been built and is now in continuous operation. The system has been useful as a stand-alone aid for debugging failures of the operational network. It provides easy access to packet traces for developers of experimental systems and protocols. The system is used mainly as a source of traces to support research in the analysis of TCP packet traces.

A study in the analysis of TCP packet traces is in progress. This study explores the graphical presentation of packet traces to a human analyst and its effect on the human's ability to absorb and understand a packet trace.

2.6 Random Drop Queue Management

The congestion control scheme currently employed in Internet gateways is a simple mechanism that requires no information about the connections passing through each gateway. In the absence of such per-connection information, it is difficult to give the connections accurate signals in the event of congestion. One hypothesis advanced within the Internet Engineering Task Force was that a slightly more intelligent, statistical mechanism might afford satisfactory results without the need for per-connection information. This mechanism was dubbed "random drop" because it entails discard of a randomly chosen packet from the bottleneck queue whenever congestion is detected. In theory, the random drop scheme discriminates against connections that consume an inordinate share of available bandwidth, for such connections are likely to have more packets in the queue than other connections, and the probability of packet loss for any one connection is proportional to the frequency of its representation in the bottleneck queue. On the strength of this analysis, random drop was expected to afford both service equity and improved aggregate performance.

By simulating the random drop mechanism, Eman Hashem found that it does not perform as well as expected and that performance improvements over the current scheme are negligible. While random drop can improve the fairness of the gateway packet drop, it has no necessary effect upon the aggregate distribution of bandwidth in the network—which is largely determined by the interaction between gateways and TCP congestion control schemes. Even though random drop penalizes the connections fairly for causing congestion, it can not control their flow. Any connection that attempts to maximize its flow is rewarded with a higher share of available bandwidth. For example, connections with short end-to-end delay react to the gateway congestion signal quickly and recover faster—thus realizing higher throughput by squeezing bandwidth away from longer delay connections. A similar bandwidth advantage is realized by misbehaving connections that use redundant transmissions to increase the probability of data reaching the destination in a minimum number of round trips. Thus, any TCP connection, well behaved or misbehaved, that has the ability to increase its flow above the other connections can achieve a higher bandwidth share even while employing random drop. Unfortunately, this behavior also degrades the performance of the other connections, for they spend much of their reduced bandwidth shares recovering from the congestion caused by the aggressive connections.

One variation on random drop is to drop packets with some small probability before the buffer of the bottleneck resource is 100% full. In this way, the connections contributing most to congestion are afforded an early signal to slow down before gateway queues begin to overflow. This scheme, dubbed "early random drop," may represent a profitable balance between dropping too early, effectively reducing the resource's utilization, and dropping too late, achieving no improvement over the simple random drop. Early random drop is still under study with no significant advantages seen so far.

2.7 Advanced Network Simulator

Previously, Andrew Heybey and David Martin, with help from other members of the group, developed a network simulator to support the research described in the previous sections. The simulator uses the X Window System to display the state of the network as the simulation is running, and to allow the user to use the mouse to change the simulation parameters. Data produced by the simulation can also be logged to disk for post-processing. The simulator, by permitting a visual display of the network behavior as the simulation proceeds, permits a quick and intuitive understanding of complex network behavior.

In this year, Andrew Heybey has improved the performance of the simulator by eliminating bottlenecks in the simulator's X Window user interface code, and by using the ability of the GNU C compiler to compile functions inline to eliminate procedure call overhead where possible. A variety of bugs have been fixed, and the improvements have been released to other interested parties, for whom at least a minimal level of support is provided. The simulator is being actively used by people at Washington, Cray, Purdue and Mitre.

2.8 Network Naming Services

Karen Sollins continued her work on providing directory services in the Internet. A directory service permits the location and identification of people, services, and resources in order to access them. A number of directory services exist, so the focus of this work is to provide a framework for accessing a directory service that is general enough that most existing directory services can add a simple access veneer, while providing a rich communication protocol for requesting information from such a service. The framework permits directory services to name each other as well, in order to navigate through a set of such services.

Work began this year on a written plan for developing and deploying directory services in the Internet, and that effort nears completion.

Sollins organized a workshop to discuss white pages directory services in the Internet. This workshop met at the Corporation for National Research Initiatives in Reston, Virginia, and involved participants from research and industry. In addition, Sollins participated in a workshop organized by NASA and DOE that addressed problems of naming entailed by a transition from the Internet to the OSI protocol suite.

2.9 Policy Routing

David Clark completed work on a proposal for policy routing in the Internet [81]. An integral component of the Internet protocols is the routing function, which determines the series of networks and gateways a packet will traverse in passing from the source to the destination. Although there have been a number of routing protocols used in the Internet, they share the idea that one route should be selected out of all available routes based on minimizing some measure of the route, such as delay. Recently, it has become important to select routes in order to restrict the use of network resources to certain classes of customers. These considerations, which are usually described as *resource policies*, are poorly enforced by the existing technology in the Internet. Clark proposes an approach to integrating policy controls into the Internet.

The proposal models the resources of the Internet (networks, links, and gateways) as being partitioned into *Administrative Regions* or *ARs*. Each AR has a globally unique name and is governed by a somewhat autonomous administration having distinct goals as to the class of customers it intends to serve, the qualities of service it intends to deliver, and the means for recovering its cost. To construct a route across the Internet, a sequence of ARs must be selected that collectively supply a path from the source to the destination. This sequence of ARs is called a *Policy Route*, or *PR*. Each AR through which a Policy Route passes will be concerned that the PR has been properly constructed, that is, each AR may wish to insure that the user of the PR is authorized, the requested quality of service is supported, and that the cost of the service can be recovered. Before a PR can be used, however, it must be reduced to more concrete terms: a series of gateways which connect the sequence of ARs. These gateways are called *Policy Gateways*.

Clark's proposal is designed to permit as wide a latitude as possible in the construction and enforcement of policies. In particular, no topological restrictions are assumed. In general, the approach is driven by the belief that, since policies reflect human concerns, the system should primarily be concerned with enforcement of policy, rather than synthesis of policy.

The proposal permits both end points and transit services to express and enforce local policy concerns.

2.10 Byzantine Routing Algorithms

Most dynamic network layer routing algorithms depend on the proper operation of all the routing nodes for their correct operation. If one node is corrupted, and for example asserts that it is the best route to all destinations, most routing algorithms fail to detect that this is an error.

Radia Perlman completed her study [258] of Byzantine routing algorithms—routing algorithms that continue to operate correctly even if one or more routing nodes are corrupted in malicious ways.

2.11 Network Management

James Davin has continued his efforts to develop the Simple Network Management Protocol (SNMP) by participation in the relevant Internet Engineering Task Force working groups, by authoring documents that specify the protocol [73][74][282] and explain its design [103], and by implementation. During this period, the MIT SNMP Development Kit software was developed and initially released. This software is a highly portable C language implementation of the SNMP and has been ported to a variety of platforms. In particular, SNMP was implemented in the MIT C Gateway as part of this effort.

2.12 Internet Architecture

As part of our research effort, and in support of the ongoing extensions and changes to the Internet protocol suite, members of the group participated in a number of working groups, and contributed a number of design papers [80].

Internet Activities Board: David Clark continued to chair the Internet Activities Board, the steering board for the Internet protocol suite. He also attended the meetings of several working groups and task forces of the IAB.

Inter-Autonomous System Routing Architecture: Lixia Zhang continued participation in the Open-Routing Working Group under the Internet Engineering Task Force.

Naming Services: As part of the work reported earlier on architectures for naming, Karen Sollins is a member of the Naming Task Force of the Distributed Systems Activities Board. She is also a member of the Autonomous Systems Task Force of the IAB.

2.13 Publications

[1] D.D. Clark, V. Jacobson, J. Romkey, and H. Salwen. An analysis of TCP overhead. *IEEE Communications Magazine*, 27(6):23–29, June 1989.

[2] D.D. Clark. *Policy Routing in Internet Protocols.* Request for Comments 1102, DDN Network Information Center, SRI International, May 1989.

[3] D.D. Clark. The design philosophy of the DARPA internet protocols. In *Proceedings of ACM SIGCOMM*, pages 106–114, Stanford, CA, August 1988.

[4] D.D. Clark and D.R. Wilson. Evolution of a model for computer integrity. In *Proceedings of the 11th National Computer Security Conference*, pages 14–27, Baltimore, MD, October 1988.

[5] M.S. Fedor, J.R. Davin, M.L. Schoffstall, and J.D. Case. The SNMP and protocol design for network management. In *Proceedings of the IEEE Systems Design and Networks Conference*, Santa Clara, CA, May 1989.

[6] J.D. Case, M.S. Fedor, M.L. Schoffstall, and J.R. Davin. *A Simple Network Management Protocol.* Request for Comments 1067, DDN Network Information Center, SRI International, August 1988.

[7] M. L. Schoffstall, J. R. Davin, M. S. Fedor, and J. D. Case. *SNMP over Ethernet.* Request for Comments 1089, DDN Network Information Center, SRI International, February 1989.

[8] J. D. Case, M. S. Fedor, M. L. Schoffstall, and J. R. Davin. *A Simple Network Management Protocol.* Request for Comments 1098, DDN Network Information Center, SRI International, April 1989.

Thesis Completed

[1] R. Perlman. *Network Layer Protocols with Byzantine Robustness.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, August 1988.

Theses in Progress

[1] E. Hashem. *Analysis of Random Drop for Gateway Congestion Control.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected 1988.

[2] T. Shepard. *An Intelligent Extensible Analyzer for TCP Packet Traces.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected 1988.

[3] L. Zhang. *A New Architecture for Packet Switching Network Protocols.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected 1988.

Lectures

[1] D. Clark. Limits on autonomy, or distributed systems considered harmful. Lecture given at ACM SIGOPS Workshop, September 1988.

[2] D. Clark. Networks—the coming revolution. Lecture given at DARPA Principal Investigators Meeting, November 1988.

[3] D. Clark. The changing nature of computer networks. Lecture given at Bellcore (March), Carnegie Mellon University (April), National Science Foundation (May), 1989.

[4] D. Tennenhouse. Layered multiplexing considered harmful. Lecture given at IFIP Workshop, Zurich, Switzerland, May 1989.

Clinical Decision Making

Academic Staff

R. Patil	P. Szolovits, Group Leader

Collaborating Investigators

M. Criscitiello, M.D., Tufts-New England Medical Center Hospital
M. Eckman, M.D., Tufts-New England Medical Center Hospital
J. Kassirer, M.D., Tufts-New England Medical Center Hospital
S. Naimi, M.D., Tufts-New England Medical Center Hospital
S. Pauker, M.D., Tufts-New England Medical Center
W. Schwartz, M.D., Tufts-New England School of Medicine
O. Senyk, M.D. Ph.D., Tufts-New England Medical Center Hospital
F. Sonnenberg, M.D., Tufts-New England Medical Center Hospital
J. Wong, M.D., Tufts-New England Medical Center Hospital

Research Staff

J. Doyle	W. Long

Graduate Students

D. Aghassi	T.-Y. Leong
D. Fogg	R. Rubsamen
S. Greenwald	T. Russ
N. Harris	T. Wu
Y. Jang	A. Yeh

Support Staff

A. Ellis

3.1 Summary

This year, we continued to work on the development of fundamental new methods for representing medical knowledge and reasoning with it, exploration of means of integrating various results into coherent research systems, and formulating methods of merging decision analytic and AI reasoning. In addition, we are planning to use part of this year to summarize our accomplishments and experiences during the period of our grant, and to plan our future research.

3.2 Plans

3.2.1 Integration

A few years ago, we hypothesized that the adoption of a uniform method of knowledge representation (based on contemporary developments in AI research) would give us an important advance in the ability to integrate various specific representation and reasoning techniques. We pursued this goal aggressively, but with considerably more frustration than we anticipated. We reviewed some of our practical difficulties with this technology late last year [141] and have continued to explore the fundamental deficiencies in current AI representation approaches that have underlain those difficulties [96].

During the next year, we plan to attack these problems by taking both a theoretical and architectural approach to the problem of rational self-government, as defined by Jon Doyle. Architecturally, Doyle and Ramesh Patil are developing the principles and structures for knowledge bases which rationally manage their knowledge and inference methods. We expect this organization for knowledge representation systems to have many advantages over the current crop of systems. Many current systems restrict the expressive power of their languages in order to gain "efficiency," but as our recent work has shown, this sort of "efficiency" defeats the most important uses of these systems. Other systems are more expressive, but are still limited by their inference methods, which apply mainly logical inference procedures in ways independent of the user's goals. This also makes for inefficiency, as these systems sometimes prevent themselves from satisfying the user's needs by wasting effort on inferences irrelevant to those needs. The goal of this study is to develop a set of architectural principles that permit design and development of new representation systems that explicitly take into account the objectives and constraints that they are to satisfy.

Complementing this, Doyle and Michael Wellman[1] are studying how to achieve rationality in the process of developing and revising plans for large distributed activities. The main focus in this work will be on developing techniques for rational distributed reason maintenance. All current reason maintenance systems carry out unbounded computations at each database cycle. Though they save effort over previous approaches to belief revision by only examining a portion of the knowledge base when effecting changes, that portion may be much or even

[1] Wellman is a former student, now collaborating from the Air Force's AFWAL, Dayton, OH.

all of the knowledge base. Thus these systems are ill-suited for real time or rationally guided operations, as they provide no way for the user to control the effort spent on a revision or the distribution of effort over time. The aim of rational distributed reason maintenance is to make revision computations as local as possible, with pursuit of possible revisions across distances or different media under the influence of the goals and circumstances of the reasoner. In the theoretical work, Doyle plans to continue development of a mathematical theory of reason maintenance and rational self-government. In addition, he and Elisha Sacks[2] plan to investigate the applicability of more techniques of modern mathematics to qualitative reasoning about physical systems.

In addition to such fundamental work on knowledge representation issues, we are also planning to develop representation schemes at a higher, more specifically medically-relevant level of detail. Inspired by the qualitative probabilistic network representation pioneered by Wellman [310], Tze-Yun Leong is developing a taxonomy of the structural aspects of a decision problem. The intent is to represent such concepts as clinical contexts, classes of therapeutic interventions, causality, and dependency. By gaining insights into the structure of clinical decisions, this exercise serves as a step toward realizing the uniform knowledge representation language for an integrated artificial intelligence and decision analysis system for medical reasoning.

This work will produce an appropriate representation for the formulation of decision problems according to classical and recent models of decision making, such as qualitative or quantitative probabilistic networks, influence diagrams, or decision trees. We plan to pursue this work in the domain of pulmonary infiltrates in AIDS patients, a field in which Frank Sonnenberg is pursuing parallel and somewhat more applied studies. We plan to take advantage of his work and thoughts, using them as a basis for the more theoretical work to be done here. In particular, in the coming year we hope to have developed a set of representation conventions that are completely adequate to describe any modeling issues that arise in the course of considering an AIDS/pulmonary infiltrate case. By looking at more clinical cases, the following complicated issues will be further explored and their implications on the proposed representation framework analyzed: contextual representation, representing multiple taxonomies, classification along multiple perspectives (in the same taxonomy), and the resulting interactions among the concepts. Theoretical formalization of the representation framework will be attempted, and the resulting expressiveness of the framework will be evaluated. We expect that the constellation of issues uncovered here will generalize to a much broader set of medical domains.

3.2.2 Fundamental Methods

We are focusing on a number of important fundamental techniques for medical reasoning: (1) an elegant and flexible formulation of diagnosis, (2) a powerful method of temporal reasoning and temporal belief maintenance, and (3) a number of interesting and critical approaches to learning in expert systems.

[2]Sacks is a former student, now on the faculty at Princeton University.

Diagnostic Reasoning

Although the diagnostic algorithms of most medical AI systems have been based on a variety of *ad hoc* computational mechanisms, recent research in the formulation of diagnostic problems across other domains of AI has identified a more systematic analysis of the bases of diagnostic reasoning. Thomas Wu is investigating *case structuring*, a sophisticated strategy for solving complex medical diagnostic problems in the broad domain of general internal medicine. In a difficult, multiple-disease medical case, a diagnostic system could help a physician greatly by computing alternative formulations of the case. Case structuring generates such alternative formulations. Instead of evoking disease solutions directly (and haphazardly) from the symptoms in the medical case, our approach introduces an intermediate clustering step to identify coherent aggregates of symptoms. Each aggregate of symptoms is caused by a separate disease and therefore represents a separate differential diagnostic task to be solved. A coherent set of tasks that explains the entire case is called a task formulation; the set of task formulations constitutes the alternative formulations of a medical case. Due to the intermediate clustering step, this method of case structuring is called *symptom clustering*.

Symptom clustering derives its computational power from two sources. First, it exploits mutual constraints that derive from the symptoms in a case. Second, it takes advantage of the dual observation that many diseases map onto a few functional derangements and that each functional derangement maps onto several co-occurring symptoms. These functional derangements—called syndromes in clinical practice—offer a powerful source of heuristic knowledge for finding the correct task formulation. In the past year, Wu has identified the case structuring strategy of medical problem solving, developed algorithms for the symptom clustering methodology, and implemented a prototype diagnostic system.

In the next year, he plans to extend the theoretical framework of case structuring, refine the prototype diagnostic system, and evaluate the results. The extensions will cover a number of broad issues in medical problem solving, including causal relationships, probabilistic assessment and test generation and problem solving strategies.

Causality in the current methodology is very shallow, allowing for only pathophysiological causality between diseases and the symptoms they can cause. However, diseases may cause other diseases, or they may synergistically enhance or oppose each other, thereby changing their symptomatology. These processes of causal predisposition and causal interaction can be incorporated into the symptom clustering methodology by modifying the knowledge base and diagnostic algorithm.

Probabilistic notions are missing in the current framework, which is unrealistic since the likelihood of diseases and causal influences ranges over several orders of magnitude. Therefore, Wu will introduce prior probabilities for diseases and causal probabilities for associations between symptoms and diseases. These probabilities will entail several considerations for research: they can be changed by contextual information, such as age, sex, race, occupation, and medical history; they can help guide the search for plausible task formulations; and they can help determine symptoms that do not need to be explained.

Test generation and problem solving strategies are needed to make the diagnostic system truly interactive. The current framework receives symptoms passively from the physician user; with test generation capabilities, it could actively seek relevant symptom data. Problem solving strategies are sequences of tests, representing the procedural knowledge that medical experts use to solve cases. Wu plans to incorporate test generation and problem solving strategies in a separate system to complement the existing diagnostic system.

A prototype diagnostic system has been implemented and tested on a small medical knowledge base. To fully test the strengths and limitations of our approach, however, we plan to use a large knowledge base developed by the INTERNIST project. The form of the INTERNIST knowledge base is suitable for this diagnostic algorithm. Evaluation of our approach will therefore be based upon empirical comparisons between case structuring and the direct approach of the INTERNIST system.

Temporal Reasoning

Thomas Russ is continuing the development of the Temporal Control Structure for creating expert systems that use time dependent data. The research has identified schemas of temporal reasoning such as the abstraction of patient states from the analysis of examination and laboratory results. Support for using hindsight to evaluate previously made decisions was also provided. A paper titled "Using Hindsight in Medical Decision Making" was accepted by the Symposium on Computer Applications in Medical Care, to be held in October 1989 [278]. It will be a finalist in the student paper competition sponsored by the Symposium. This programming methodology and the system built to support it appears especially effective for the implementation of monitoring and tracking systems. Russ, as part of his doctoral dissertation, is using the system to implement a system for monitoring the treatment of patients with diabetic ketoacidosis. We expect this work to be completed by the end of 1989.

A different problem of temporal reasoning arises in the Heart Failure (HF) Program. One of the problems with the existing Heart Failure Program is the inability to deal with situations in which the order of events or the time between cause and effect is important. Part of the reason for this difficulty is that the knowledge base representation of cause and effect does not include the properties needed for reasoning about the time relations. During the past year we have been developing a representation for such time relations that will allow specifying the essential features without requiring more specificity than is possible. The kinds of distinctions needed include the time required to produce an effect, the duration of an effect after the cause has been removed, the type of onset (gradual or acute), and the nature of findings (sampled versus symptoms with duration). Since the probability of the effect is often a function of both the duration and the severity of the cause, suitable approximations of these functions need to be part of the knowledge base. We have been developing a representation that will capture these relations allowing reasoning with hypotheses that are clinically distinct even though they involve the same nodes. We used the representation for an example knowledge base that captures the important clinical distinctions in a case that the existing Heart Failure Program is unable to handle. We are currently working on strategies for limiting hypotheses to only those that are different in significant ways.

Learning

A number of factors have come together to suggest the critical and ever-more-widely recognized role of learning in medical AI. First, it is widely recognized that the handcrafting of expert systems is a difficult and time consuming task that could be significantly eased if part of model construction could be automated. Second, the increasing availability of large bodies of relatively carefully collected real clinical data means that system builders need not rely on human expert judgment as exclusivelv as we once did. We are investigating the applicability of several learning approaches.

In the context of the Heart Failure Program, we are comparing machine learning approaches, such as that of ID3 [268], for producing decision trees to the logistic regression approach used on a large database of patients presenting with chest pain [267]. We made arrangements with Drs. Selker and D'Agostino [267] to use their database of 5773 cases with chest pain or shortness of breath in the emergency room. This database has many clinical attributes as well as the primary final diagnosis. It was used to develop a predictive instrument using logistic regression analysis to determine the probability that a patient has acute cardiac ischemia. Because of the care with which the data was collected and the large amount of data collected on each patient, this is an ideal database for comparing other technologies to that of logistic regression analysis. Our first step is to use the machine learning program ID3 to explore the kinds of decision trees that it will generate with the same data. ID3 is representative of a class of machine learning programs that inductively generate decision trees from examples using statistical tests and heuristics to keep the trees small and limited to only those categorizations that are statistically justified. Since these technologies have been enhanced to handle noisy and missing data, they are capable of handling a database such as this one. We have a working implementation of ID3 (implemented at our institution by a graduate student, Jonathan Amsterdam) which has been tested on a number of small datasets. We are currently checking the data and designing experiments leading up to running ID3 on the whole learning set later in the summer.

Yeona Jang plans to conduct investigations (eventually leading to her Ph.D. thesis) on how to learn clinical rules in a medical reasoning program that incorporates both associational and causal knowledge. The primary goal is to create a system that can learn from its experiences (in particular, to be able to use an indication of whether its conclusions were correct (or acceptable) to critique and revise its knowledge and decision methods.

David Aghassi intends to continue his study of the applicability of formal, algorithmic learning methods to problems of medical knowledge.

Intelligent Signal Analysis

Patil and Scott Greenwald have been investigating the use of AI methods to assist in the interpretation of physiologic signals, in particular the design and testing of algorithms that automatically analyze two leads of the electrocardiogram. The major emphasis of this work has been in the continued development of CALVIN, an expert system that exploits contextual information in the ECG. CALVIN has been developed to enhance the detection of normal beats and isolated ventricular premature beats in the presence of severe electrode motion noise and QRS-like artifact. In addition, we also modified ARISTOTLE, a traditional arrhythmia detection algorithm. Greenwald has developed and evaluated a noise detection strategy using the ratio of the number of peaks to beats detected within a three second window. A ratio greater than a threshold signifies the presence of severe noise.

In the last year, the results of CALVIN's detection performance were presented at the Computers in Cardiology Conference (September 1988) [138]. In addition, the work was discussed at the Association for the Advancement of Medical Instrumentation Conference (May 1989). Finally, we plan to present a working real time demonstration of the combined ARISTOTLE/CALVIN arrhythmia analysis system on a MAC II at the 1989 Computers in Cardiology Conference (September 1989.)

The future direction of this work is focused on improving the detection of isolated atrial premature beats (S), isolated premature ventricular beats (V), and normal beats (N) in the presence of severe noise. The following projects will help bring that goal closer.

ECG Database Development: A database containing sinus arrhythmia and atrial ectopic activity needs to be created for developing and testing CALVIN's performance on atrial arrhythmias. A database consisting of twenty 1/2 hour sections (10 for detector development and 10 for detector testing) will be collected for each of the following arrhythmia classes:

1. isolated atrial premature beats;
2. atrial couplets, triplets, runs of atrial tachycardia, and paroxysmal atrial fibrillation;
3. mixed isolated atrial premature beats and isolated ventricular premature beats;
4. mixed atrial and ventricular couplets, triplets, and tachycardia;
5. real-world noise (primarily electrode motion artifact); and
6. sinus arrhythmia (normals will need to be collected from Beth Israel Hospital Holter Laboratory).

Improving CALVIN's Performance: The current CALVIN system works well in correcting ARISTOTLE's errors in classifying normal beats and isolated premature ventricular beats in normal sinus rhythm. However, it continues to make mistakes matching hypothetical sequences of beats to the raw data in two cases: 1) in regions where the heart rate is moderately increasing or decreasing (say over 5 or so beats), and 2) in regions where the heart rate is constant but where its estimate of the heart rate is inaccurate.

One possible solution to this problem is to improve our estimate of heart rate (using beats classified with high confidence within a wide context) and to "stretch" the hypothetical sequences of beats in a nonlinear fashion to account for heart rate variations. The following two projects would help us meet this goal.

One important question to answer is how interbeat intervals vary as a function of heart rate. For example, normal-to-ventricular beat coupling intervals (NV intervals) tend to be constant to a first approximation as heart rate varies. We need to collect statistics on the relation of NS, SN, NV, and VN intervals as a function of heart rate in order to determine how to "stretch" and match hypothetical sequences of beats for a particular heart rate.

During periods of noisy ECG, the noise level may drop momentarily. During this brief period one or two heart beats may become quite apparent. Physicians frequently use these "landmark" beats as fidicual points to re-adjust their estimates of heart rate and their expectations of beat locations.

We need to develop a method to find these landmark beats, and to use them in improving our heart rate estimate and in selecting the correct hypothetical sequences of beats. One possible confidence measure to use to find these landmark events is the correlation coefficient of the data with the best matched beat template. If the correlation were above a conservative threshold, then our confidence in the beat's identity would be high enough to update our heart rate estimator.

3.2.3 Probabilities and AI

As part of our strong ongoing interest in the integration of probabilistic and artificial intelligence methods, we are pursuing—and plan to continue to pursue—a number of related issues. Fundamental to all this work is the question: what is the most appropriate form in which to capture and represent probabilistic information in such a way that it is useful to physicians?

Computations in Probabilistic Networks

Research over the past few years on the evaluation of Bayesian probability networks has led to the development of new algorithms for handling multiple paths between nodes. In particular, the approach of Lauritzen and Spiegelhalter [196], while not circumventing the inherent exponential nature of the problem, seems to be significantly faster than other published approaches. This approach should be fast enough to handle networks with tens of nodes and several multiple paths. We are implementing the algorithm to determine its practical limitations and to apply it to problems of about that order of complexity. The speed of this algorithm should also make it possible to test the heuristic approach we are using on larger networks. The empirical tests we conducted over the past two years on the heart failure program by entering actual cases and critiquing the differential diagnoses have convinced us that our current heuristic approach to the evaluation of the network (which has about 150 intermediate nodes, about 300 possible terminal nodes, many multiple paths, and even forward loops) produces good hypotheses [216]. By inspection we have not found better hypotheses than the ones produced by the heuristic method, but we have not had an exact method for determining the best hypotheses. Our implementation of the Lauritzen and Spiegelhalter algorithm will allow us to test the performance of our heuristic algorithm on somewhat simplified models. Investigation of the Lauritzen and Spiegelhalter algorithm has taken place and we are beginning the implementation, which should be completed shortly.

Probabilistic Reasoning for Genetic Pedigrees

We plan to continue a collaborative effort principally undertaken during the past year with Susan Pauker, responsible for the clinical genetics counseling program of the Harvard Community Health Plan, and her genetic counseling colleagues. Pauker has a longstanding interest in the application of probabilistic reasoning methods to her counseling practice [255][256]. Nomi Harris, as part of her Master's thesis [155], has completed a program that employs the techniques of Bayes networks [257] to solve the probabilities of a consultant's risk of abnormality for arbitrary pedigrees, including cases of inbreeding. The program handles diseases that may be dominant, recessive or sex-linked, and has facilities for dealing with incomplete penetrance, mutation, time-varying likelihood of expression, etc. A paper describing this work has been accepted for the upcoming SCAMC meeting and is also a finalist in the student paper competition [154].

We now ported the computational core of that program to personal computers (the Macintosh and a fully equipped 386-based PC), and are investigating the design of appropriate user interfaces. In addition, we plan to put this program into the hands of practicing genetic counselors within the next few months asking them to perform retrospective evaluations of some of their clinical cases and, based on feedback from their experience, outline what additional capabilities are necessary to support the analyses performed by these counselors. We also plan to explore the adoption of the Spiegelhalter and Lauritzen network algorithm to speed up the program on complex, interbred pedigrees.

Extracting Probabilistic Information from Partial Databases

We are also using the Selker and D'Agostino database (described above under "Learning") to investigate the probabilities in the Heart Failure Program. Since many clinical attributes of the patients were collected, it is possible to identify subsets of the database that have properties corresponding to some of the intermediate nodes in the model and determine the probability and confidence intervals for some of the findings given those states. It is not possible to do this for all of the nodes in the model because not all of the data was collected, some of the findings in the model do not exactly correspond to the data collected in the database, and no secondary diagnoses were recorded in the database. Even so, the data gives us an opportunity to compare the actual occurrence of some findings in the model to the probabilities given by our cardiology experts from their clinical experience and knowledge of the literature. We are currently determining the correspondence between the model nodes and the database attributes and determining which relations can be tested.

Decision Tree Critiquer

In the past year, we began a study of the efficacy of the decision tree critiquer (DTC) in detecting structural errors in decision trees. Recall that DTC was designed after first observing the kinds of errors that arose in trees built by trainees on the clinical decision making consultations service. In this ongoing study, initial trees presented by trainees and students are captured and translated from the PC environment to the symbolic environment by a neutral research assistant (a medical student). The critiquing program then lists the potential structural errors identified by its knowledge base. Because of the annual turnover of fellows and more frequent turnover of students and visitors to the service, we anticipated that the frequency of errors identified would be of the same order of magnitude as was found two years ago when we initially developed our error catalog and began to code DTC.

To our surprise, we have been identifying far fewer structural errors, suggesting that the educational process has been passed from trainee to trainee. At this point most of the "problems" identified by DTC are not true errors but rather represent instances in which relationships that might have been represented structurally have been incorporated into "binding expressions" within the decision trees. Such expressions are essentially microcoded programs that the current implementation of DTC cannot parse into relations among meaning concepts. We feel that this represents more than just a limitation in DTC. When decision trees are used by the clinical service, they undergo an extensive debugging process, which in fact almost uniformly identifies problems and necessary refinements. Those problems, as identified by human experts (faculty), almost always lie within expressions within the "bindings," probabilities, and utilities of a decision tree. This concordance of error locations suggests that our current classic formalism for problem representation, which hides certain relationships within algebraic expressions, is inadequate.

One ready hypothesis might be that decision trees are a less adequate representation than influence diagrams of complex decision problems. In fact, the two formalisms appear to be complementary representations, with each making explicit certain relations that are "hidden" in the other—in the tables of the influence diagram and in the binding structure of the decision tree.

In the next year, we plan to allow DTC to process the decision trees created by a new crop of fellows (five new to our division, four new to decision analysis). We also plan to process a random selection of trees from the published literature, omitting trees created by our division or by former trainees. Unfortunately, the current DTC implementation is incapable of examining the details of the model contained within mathematical expressions (utilities, probabilities, bindings) and only critiques the structure of the decision tree. We plan to expand this system, developing a syntax and rules for examining the content of expressions and the structure of Markov components. The examination of the content of expressions first requires that a consistent formalism be developed for representing the semantics contained within those expressions, creating appropriate links. Eventually, one would like to be able to link those expressions to a domain-specific knowledge base, but we will first explore the nature of relations among variables without reference to their medical content.

We already began an implementation of the existing DTC in the same microcomputer environment as DecisionMaker, using the Goldworks systems. We hope to complete that implementation in the next six months and begin to examine the feasibility of its use concurrent with tree development.

Microcomputer Implementations

Substantial portions of the decision tree critiquing system have been ported to the IBM environment in the Goldworks system. That partial implementation is functioning, and we are exploring mechanisms for allowing it to interact conveniently with the evolving DecisionMaker environment.

DecisionMaker itself has been expanded to include more transparent representation of subtrees, and we have been exploring alternate visual representations of the implicit information contained within bindings. We also improved the scripting mechanism and completed the implementation of Monte Carlo simulations within our microcomputer environment.

DecisionMaker is now a fairly stable product in a Pascal environment on the IBM compatible computer family. We plan to explore the feasibility of a direct port to the Macintosh world using the Borland Pascal environment. We also plan to use the new Borland object-oriented programming modules within their Pascal system to implement a portion of the DTC (now on Symbolics and Goldworks) system, and provide concurrent advice on tree construction and the interpretation of the results of sensitivity analyses.

Applications

We have been developing an extensive model comparing coronary bypass surgery, percutaneous transluminal angioplasty, and conservative therapy in patients with angina. That complex Markov model required expansion of the capabilities of our modeling environment which has now been completed. The results of that cost-effectiveness analysis are now under review. We also developed models of screening for sickle cell disease, the selective use of cytomegalovirus immune globulin in renal transplant recipients, the expanded use of thrombolytic therapy in patients with acute myocardial ischemia, and the determination of occupational risk of HIV infection.

3.3 Publications

[1] J. Doyle. Constructive belief and rational representation. *Computational Intelligence*, 5(1):1–11, February 1989.

[2] J. Doyle. Mental constitutions and limited rationality. In M. Fehling and S. Russell, editors, *Papers of the AAAI Symposium on AI and Limited Rationality*, 1989.

[3] J. Doyle. Reasoning, representation, and rational self-government. In Z. W. Ras, editor, *Methodologies for Intelligent Systems, 4*, pages 367–380, North-Holland, 1989.

[4] J. Doyle and R. S. Patil. *Language Restrictions, Taxonomic Classifications, and the Utility of Representation Services.* Technical Report MIT/LCS/TM 387, Laboratory for Computer Science, May 1989.

[5] J. Doyle and E. P. Sacks. Stochastic analysis of qualitative dynamics. In N. S. Sridharan, editor, *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1187–1192, Morgan Kaufmann, 1989.

[6] J. Doyle and M. P. Wellman. Impediments to universal preference-based default theories. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 94–102, Morgan Kaufmann, May 1989.

[7] M. H. Eckman. A counterpoint to the analytic hierarchy process. *Medical Decision Making*, 9:57–58, 1989.

[8] M. H. Eckman, J. R. Beshansky, I. Durand-Zaleski, H. J. Levine, and S. G. Pauker. Peri-operative management of anticoagulation in patients with prosthetic heart valves undergoing non-cardiac surgery. *Medical Decision Making*, 8:328, 1988. Presented at the Society for Medical Decision Making, Tenth Annual Meeting, Richmond, VA, October 17–19, 1988.

[9] M. H. Eckman and S. G. Pauker. Principles of diagnostic testing. In W. Kelly, editor, *Textbook of Medicine*, J. B. Lippincott, 1989.

[10] L. Ellwein and M. H. Eckman. Biological complexity in mathematic modeling. *Medical Decision Making*, 9:38–39, 1989.

[11] J. E. Gottlieb and S. G. Pauker. When is a positive PPD falsely positive? A new look at old tuberculin. *Seminars in Respiratory Medicine*, 10:218–226, 1989.

[12] S. D. Greenwald, R. S. Patil, and R. G. Mark. Improved arrhythmia detection in noisy ECGs through the use of expert systems. In *Computers in Cardiology*, 1988.

[13] M. Hagen, K. Meyer, and S. G. Pauker. Human immunodeficiency virus infection in health care workers: a method for estimating individual occupational risk. *Archives of Internal Medicine*, 149:1541–1544, 1989.

[14] M. D. Hagen, M. H. Eckman, and S. G. Pauker. Aortic aneurysm in a 74-year-old man with coronary disease and obstructive lung disease: is double jeopardy enough? *Medical Decision Making*, 9(4):285–299, October-December 1989.

[15] M. D. Hagen, M. H. Eckman, and S. G. Pauker. The decision problem and the decision model: choose the right tool for the job. *Medical Decision Making*, 9(4):325, October-December 1989. Presented at the Society for Medical Decision Making, Eleventh Annual Meeting.

[16] I. J. Haimowitz, R. S. Patil, and P. Szolovits. Representing medical knowledge in a terminological language is difficult. In *Symposium on Computer Applications in Medical Care*, pages 101–105, 1988.

[17] D. E. Hirsch, S. R. Simon, T. Bylander, M. A. Weintraub, and P. Szolovits. Using causal reasoning in gait analysis. *Applied Artificial Intelligence*, 3(2-3):337–356, 1989.

[18] R. L. Jayes, N. S. Hill, and S. G. Pauker. Open lung biopsy in primary pulmonary hypertension: a decision analysis. *Seminars in Respiratory Medicine*, 10:232–241, 1989.

[19] J. P. Kassirer and R. I. Kopelman. Cognitive errors in diagnosis: instantiation, classification, and consequences. *American Journal of Medicine*, 86:433–441, 1989.

[20] J. P. Kassirer and F. A. Sonnenberg. Clinical decision analysis. In W. N. Kelley, editor, *Textbook of internal medicine*, pages 30–34, J. B. Lippincott, 1989.

[21] J. P. Kassirer and F. A. Sonnenberg. The scientific basis of diagnosis. In W. N. Kelley, editor, *Textbook of internal medicine*, pages 14–16, J. B. Lippincott Co., 1989.

[22] R. I. Kopelman, R. A. McNutt, and S. G. Pauker. Use of decision analysis in a complicated case of renovascular hypertension. *Hypertension*, 12:611–619, December 1988.

[23] P. A. Koton. A medical reasoning program that improves with experience. In *Symposium on Computer Applications in Medical Care*, 1988.

[24] E. J. Lamb, M. Hagen, and S. G. Pauker. The mean interval to conception: a measure of utility for the analysis of decisions involving fertility. *American Journal of Obstetrics and Gynecology*, 160:1470–1478, 1989.

[25] W. J. Long. Medical diagnosis using a probabilistic causal network. *Applied Artificial Intelligence*, 3(2-3):367–384, 1989.

[26] W. J. Long, S. Naimi, M. Criscitiello, and G. Larsen. Differential diagnosis generation from a causal network with probabilities. In *Computers in Cardiology Conference*, IEEE, 1988.

[27] W. J. Long, S. Naimi, M. G. Criscitiello, and S. Kurzrok. Reasoning about therapy from a physiological model. *Biomedical Measurement Informatics and Control*, 2(1):40–45, 1988.

[28] A. J. Moskowitz, B. Kuipers, and J. P. Kassirer. Dealing with uncertainty, risk and tradeoffs: a cognitive science approach. *Annals of Internal Medicine*, 108(3):435–449, 1988.

[29] A. J. Moskowitz and S. G. Pauker. A patient with new Q waves: methods for decision making in the individual patient. *J. Am. Coll. Card.*, 14 (Supplement A):29A–37A, 1989.

[30] S. G. Pauker and R. I. Kopelman. Screening for renovascular hypertension: a which hunt. *Hypertension*, 14:258-260, 1989.

[31] A. Porath, J. B. Wong, H. P. Selker, and S. G. Pauker. Thrombolytic therapy for suspected myocardial infarction: a decision analytic model. In *Proceedings of Computers in Cardiology*, IEEE Computer Society Press, 1989.

[32] E. P. Sacks. Piecewise linear abstraction of intractable dynamic systems. *International Journal of Artificial Intelligence in Engineering*, July 1988.

[33] H. P. Selker, J. R. Beshansky, S. G. Pauker, and J. P. Kassirer. The epidemiology of delays in a teaching hospital: the development and use of a tool that detects unnecessary hospital days. *Medical Care*, 27:112–129, 1989.

[34] O. Senyk, R. S. Patil, and F. A. Sonnenberg. Systematic knowledge base design for medical diagnosis. *Applied Artificial Intelligence*, 3(2-3):249–274, 1989.

[35] S. R. Simon, M. A. Weintraub, T. Bylander, D. Hirsch, and P. Szolovits. Dr. Gait: an expert system for gait analysis. In *RESNA 12th Annual Conference*, Rehabilitation Engineering Society of North America, 1989.

[36] F. A. Sonnenberg, J. B. Wong, and S. G. Pauker. Modeling time-dependent parameters with a variable starting point in a Markov cohort simulation with a limited memory. *Medical Decision Making*, 8:342, 1988. Presented at the Society for Medical Decision Making, Tenth Annual Meeting, Richmond, VA, October 17-19, 1988.

[37] J. Tsevat, I. Durand, and S. G. Pauker. Antibiotic prophylaxis for dental procedures in patients with artificial joints: worthwhile, by the skin of their teeth. *American Journal of Public Health*, 79:739–743, 1989.

[38] J. Tsevat, I. Durand-Zaleski, D. R. Snydman, S. G. Pauker, B. G. Werner, and A. S. Levey. Which renal transplant patients should receive cytomegalovirus immune globulin? A cost-effectiveness analysis. *Medical Decision Making*, 8:344, 1988. Presented at the Society for Medical Decision Making, Tenth Annual Meeting, Richmond, VA, October 17-19, 1988.

[39] J. Tsevat, M. H. Eckman, R. A. McNutt, and S. G. Pauker. Quality of life considerations in anticoagulant therapy for patients with dilated cardiomyopathy. *Medical Decision Making*, 8:342, 1988. Presented at the Society for Medical Decision Making, Tenth Annual Meeting, Richmond, VA, October 17-19, 1988.

[40] J. Tsevat, W. C. Taylor, J. Wong, and S. G. Pauker. Isoniazid for the tuberculin reactor: take it or leave it. *American Review of Respiratory Diseases*, 137:215–220, 1988.

[41] M. P. Wellman. Critiquing therapy plans by incremental improvement. In *AAAI Spring Symposium on Artificial Intelligence in Medicine*, pages 101–102, 1988. Extended Abstract.

[42] M. P. Wellman. Review of Perry L. Miller, *Expert Critiquing Systems*. *Artificial Intelligence*, 35:273–276, 1988.

[43] M. P. Wellman, M. H. Eckman, C. Fleming, S. L. Marshall, F. A. Sonnenberg, and S. G. Pauker. Automated critiquing of medical decision trees. *Medical Decision Making*, 9:272–284, 1989.

Theses in Progress

[1] D. Aghassi. *Evaluating Case-based Reasoning for Heart Failure Diagnosis.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected May 1990.

[2] D. Fogg. *Artificial Intelligence and Optimization Solutions to Multi-criteria Operator Selection.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected August 1990.

[3] S. Greenwald. *Improved Detection and Classification of Arrhythmias in Noise-corrupted Electrocardiograms Using Contextual Information.* PhD thesis, Harvard University and MIT Division of Health Sciences and Technology, expected May 1990.

[4] N. Harris. *Probabilistic Belief Networks for Genetic Counseling.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected May 1990.

[5] T-Y. Leong. *Knowledge Representation for Supporting Decision Model Formulation in Medicine.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected August 1990.

[6] T.A. Russ. *Reasoning with Time Dependent Data.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected August 1990.

[7] T. Wu. *Medical Diagnostic Problem Solving Using Multiple Types of Knowledge.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected August 1990.

[8] A. Yeh. *Automatically Summarizing Repetitive Actions and Handling Parameter Uncertainty in Systems at a Steady-state.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected August 1990.

Computer Architecture Group

Academic Staff

A. Agarwal	S. Ward, Group Leader

Research Staff

M. Singh	R. Zak
R. Jenez	

Graduate Students

A. Ayers	B-H. Lim
C. Barclay	G. Maa
T. Bloomstein	J. Morrison
D. Chaiken	J. Nguyen
M. Cherian	D. Nussbaum
J. Elsbree	J. Pezaris
H. Houh	M. Powell
K. Kurihara	E. Puckett
T. King	C. Selvidge
S. Kommrusch	

Undergraduate Students

E. Anderson	N. Osgood
B. Dennis	D. Rho
M. Huffman	M. Roberts
D. Jedlinsky	R. Stata
S. Lee	T. Steele
P. Pilotte	N. Tarr

Support Staff

J. Bernard	S. Thomas

4.1 Introduction

The 1988-89 year marks the end of the Real Time Systems Group's 15-year history. RTS merged with the groups of Dally, Agarwal, and Knight to form the new Computer Architecture Group, aimed at integrating the ideas, research, and resources of a currently fragmented community.

The past year has seen substantial contraction of the RTS Group, with the departures of Robert Zak and Milan Minsky, along with the promotion of John Pezaris from staff to student. Previously reported research involving the development of the set-associative DRAM was brought to an orderly finish. The L Project continued as a focus of the group's efforts, and the seeds were planted for a new effort to develop a high performance communication and packaging substrate for chip-level digital modules. Each of these projects is the subject of a following section.

In Alewife, research has focused on large scale computer architecture and parallel processing software. The design of ALEWIFE, a scalable cache-coherent multiprocessor, is the vehicle for much of our research and is a collaborative effort with Tom Knight of the Artificial Intelligence Laboratory. The ALEWIFE multiprocessor will support multiple models of computation including shared-memory, message passing and the data parallel model.

4.2 The L Architecture

Continuing research on the L architecture (by Ayers, Minsky, Jenez, Kommrusch, Puckett, Nguyen, Pezaris, Ward, and others) led to considerable progress despite relative disinterest on the part of potential funding agencies. Recent efforts have focused on the interface aspect of L, promoting its viability as a hardware-independent virtual machine semantics rather than on the specifics of any single hardware implementation.

4.2.1 Macintosh L Implementation

A second implementation of L was made operational on the 68020-based Macintosh II, providing (1) a very different hardware platform than our previous re-microcoded Explorer, (2) an evaluation basis for L implementations on conventional processors, and (3) an environment for the development of transparent trap-and-translate software.

Rather than directly executing L instructions, the Macintosh implementation of L traps when L code is encountered and transparently translates it to native 68020 instructions. The block of native code is cached in local storage, avoiding retranslation of active program modules.

The 68020 implementation uses local RAM as a cache for active chunks, following the scheme developed by Ayers in 1987. This local memory serves both as a cache and as a site for the highest volatility level of the homogeneous storage model of L. In our current (single-processor) Mac implementation, references to non-resident chunks (*missing chunk faults*) result in access to a disk-resident external chunk space. External chunk references are patched (by the fault handler) to refer to local names as each chunk is imported, avoiding the time and cost overheads associated with conventional memory management hardware. The current scheme amounts to chunk-based virtual memory on the Macintosh, although the code anticipates sharing of the external chunk space by several L processors each having local memory.

4.2.2 Binary Compatibiiity among Inhomogeneous Machines

Our second L impiementation sheds light on a number of interesting interface issues. Our goal has been to establish the illusion of absolute binary compatibility among dissimilar machines, without imposing the overhead of interpretive mechanism. We have been able to demonstrate such compatibility using simple ("toy") programs in recent months.

Our demonstration involves starting a small L program running on one system (e.g., the Explorer); asynchronously interrupting it at some arbitrary point in its execution; copying the entire network of chunks representing the computation (including data, program, and program state) to a dissimilar machine (e.g., the Macintosh); and continuing execution without losing information or consistency. Moreover, the program runs at full native-code speeds on both machines.

4.2.3 Types as Approximations

The L base language, roughly a variant of SCHEME with compiler- rather than interpreter-based semantics, allows most references to type information to be resolved at compile time rather than incurring runtime overhead. The L compiler attempts to provide a generality/cost/performance continuum between the extremes of (say) Lisp and C by providing for runtime types but attaching a nonzero cost to their (optional) use, rewarding the author of a C-like L program by higher performance than that of an equivalent SCHEME-like one

In addition to encouraging explicit type declaration by the programmer, this compiler flexibility places a premium on mechanisms for automatic type inference. Work in this area by Nguyen and Ward has led to an interesting alternative to the mechanisms of Milner and others; our type system promises, in addition to new inference possibilities, improved type support for such language features as polymorphic functions, subtypes, and side effects.

Our type system views types as compile-time approximations of runtime values. An expression or variable may have a range of types, corresponding to varying amounts of partial information which can be inferred. The most informative type of a value is the value itself. while the least is the type *any* which applies to all values. The type of a procedural object is itself a procedure which maps types of the object's inputs to types of its outputs, whence (again) an L procedure is its own most accurate type.

The proposed inference algorithm evaluates type expressions whenever possible during compilation. As a special case, this scheme causes complete compile-time evaluation of simple functional subexpressions. However, this evaluation is complicated by (1) the possibility of non-termination (since the type language is Turing universal), and (2) the presence of side-effects (since the evaluation scheme assumes a simple applicative semantics). Under various circumstances, including timeouts and assignments, the inference scheme will revert to a weaker approximation as the type of an expression. Thus if x is the target of two assignments of types 5 and 2.718, or alternatively of the weaker types *integer* and *real*, the algorithm will infer a type for x such as *any*, *number*, or (*union 5 2.718*). The use of approximations allows us to guarantee (1) termination of the inference algorithm, and (2) functionality in the type domain. It sacrifices, of course, any hope of completeness claims for our system: our compiler will fail to discover certain inferrable types if it is forced to discard information in its approximations.

4.2.4 Cartesian Network Relative Addressing

Unlike conventional machine architectures, the programming model for L imposes no bound on the size of its addressable universe. To exploit this flexibility, work by Morrison has explored memory systems based on an addressing scheme called Cartesian Network-Relative Addressing (CNRA).

The CNRA architecture attempts to maximize scalability by using a novel addressing technique that provides some of the advantages of both global shared memory models and "local" non-shared memory models. This addressing technique assumes that the multiprocessor is built with a *direct* intercommunication network. Addresses in the CNRA system are composed of a "routing" component and a "memory location" component. The routing component indicates a path through the interconnection network. (The origin of the path is the node on which the address resides.) The memory location component is the memory location to be addressed on the node indicated by the routing component.

This addressing system offers the unlimited address space provided by local non-shared memory models, but allows easy sharing of data structures in the style permitted by global shared memory machines. The thesis discusses how a practical CNRA system might be built. There are discussions on how the system software might manage the "relative pointers" in a clean, transparent way; solutions to the problem of testing pointer equality; protocols and algorithms for migrating objects to maximize communication locality; garbage collection techniques; and other aspects of the CNRA system design. It is clear that the CNRA system is scalable (in terms of demonstrating a way to connect many processors). However, whether or not the system will work well will depend on the communication behav'our of large multiprocessor programs. Since this is not yet well understood, simulation will be required to test the viability of the CNRA architecture.

4.3 NuMesh

This section reports very preliminary thoughts on a proposed new research effort involving Ward, Dally, Agarwal, Knight, and others. The idea is currently in its fetal stage.

4.3.1 Introduction

Over the past two decades, the backplane bus has dominated computer architectures as the mechanism for intermodule communications. The reasons for this dominance are simple and remain compelling: a well-designed bus provides a simple, extensible communications substrate which allows modules performing a variety of computational tasks (and produced by a variety of manufacturers) to be assembled into coherent systems. It induces a Tinkertoy-set modularity at the system configuration level, allowing system designers to build systems without redesigning every component.

The technical limitations of buses are well known, however. Since they serialize all system-level communications, they constitute a communication bottleneck, and one whose capacity remains roughly constant as the system size is increased. Moreover, the timing of a bus is constrained by a fundamental space/time tradeoff: the time taken by each transaction must accommodate the physical length of the bus, ensuring a relatively low communication bandwidth on all but trivially short buses. Other overhead, such as the need to arbitrate the shared communication resource among competing requests, further reduce the viability of the bus as a basis for communication in high performance systems.

The following paragraphs propose the development of a communications substrate which affords Tinkertoy-set modularity and very high performance communication in a constrained but interesting range of applications. The approach involves standardizing the mechanical, electrical, and logical interconnect among modules arranged in a (partially populated) 3D mesh whose lowest level communications follow pre-compiled systolic patterns. The attractiveness of the scheme derives from the separation of its communications and processing components, and the standardization of the interface between them. This decoupling of computation from the communication substrate allows the continued exploitation of mass-produced processing elements, e.g., contemporary signal processing chips.

The goal is a set of hardware modules and support software which allows high performance, special purpose multiprocessors to be configured for particular applications in a matter of hours. Applicability of processors so configured will be restricted to relatively static, limited-connectivity algorithms such as those found in signal processing, graphics, or other real time applications; this proposal does not address the problem of general purpose multiprocessing (for which alternative proposals abound). It has the potential, however, of delivering economical supercomputer power in an interesting but limited set of application domains.

4.3.2 Modules

Each component of our system is a computational module, perhaps occupying a two-inch cube (or conceivably much smaller). Each module contains common circuitry devoted to low level communications and control functions, as well as one or more off-the-shelf chips which perform computation. Initially, we envision modules containing a modern 60-MFlop DSP and (say) a modicum of additional memory; eventually, a repertoire of compatible modules offering varied functionality (sharing the communications circuitry) might evolve.

The common circuitry (likely one or several ASICs) includes communication/control processor and a number of receiver/transmitter pairs which drive lines to neighboring modules. The communication ports (and accompanying mechanics, connector technology, etc.) allow the modules to be configured into a limited-connectivity network; our preference is a 6-neighbor 3D mesh, although other topologies are of course possible. Operation of the entire array is synchronized to a fast clock, whose period is the minimum time necessary to transfer a data word (32 bits?) between adjacent nodes. The limited distances (an inch) and fixed mechanics should allow this time to be quite fast, perhaps 10 nanoseconds or less. A variety of other functions, such as trimming clock skew, intermodule control, and processor/communications synchronization, are also performed by the common circuitry.

Initial prototypes will undoubtedly use pre-packaged off-the-shelf chips for the function-specific portion of each module. Assuming wild success of the NuMesh and bandwagon momentum among suppliers of high performance silicon, however, one might imagine each manufacturer packaging and bonding chips directly into a pre-constructed NuMesh package.

4.3.3 Communications

Each communications processor consists of a simple FSM which follows a periodic sequence of I/O transactions. It has a small number of registers, each of which is addressable by the DSP as an external memory location. The transition table of the FSM (in RAM) can be programmed to read inputs from various neighbors into registers and send outputs from various registers to other neighbors on each clock cycle. In the most ambitious configuration, any port may be read or written (or perhaps both, since we presume the lines to be unidirectional) on each clock cycle. Implementation considerations may dictate further restrictions; e.g., allowing only one output datum (perhaps to several destinations) and one input datum per clock cycle. The latter restriction is suggested by an implementation involving internal input and output buses interconnecting separate receiver/transmitter chips for each port. A myriad of other compromises are possible, depending on resources at the technological level.

The general idea is that each module's communications FSM be programmed to follow a periodic pattern of interactions with neighbors. Although the interactions may vary among processors, the periods will be identical. If module A transfers a word to its right-hand neighbor B on clock 37 of each period, then A's FSM will be programmed to drive its lines to B on that clock, while B will be programmed to load in data from A. By appropriate design of transition tables, arbitrary systolic communication patterns may be implemented among processors. In some cases, words loaded by a module are destined to be read subsequently by that module's DSP; in other cases, they are routed (typically on the next clock) to another neighbor without DSP intervention or even awareness.

Certain algorithms may benefit from flow control and other synchronization measures in their underlying communications. These might be superimposed on the primitive (branch-free) communication mechanism by software convention, allowing certain data words to contain control information. It is possible that analysis of potential application code will suggest the addition of hardware support for such control purposes.

Additional protocol provisions allow the communications FSMs, and perhaps the functional circuitry attached to them, to be programmed. This process is viewed primarily as a bootstrapping operation, and may be relatively slow; however, the potential for time-varying communication patterns may eventually be explored.

4.3.4 Software

The rapid prototyping of ad hoc multiprocessors depends on automation of various aspects of the design task, including (1) design of the network topology, (2) allocation of computational tasks to processors, (3) specification of details regarding timing and direction of communications for each module/clock pair, and (4) programming of the DSP. While steps (1) and (2) are the most challenging, they are amenable to partial solutions (e.g., involving interaction and direction from the designer) and benefit enormously from the restriction to static algorithms with time- and space-bounded components.

Code generation involves an accurate, detailed model of DSP timing—perhaps including cache operation. However, hardware provisions (e.g., bit/register R/W sync bits, like I-structures) might provide some timing latitude in DSP-FSM synchronization. Placement and routing aspects of system design—mapping a graph of time-bounded computations to a grid of processors—can probably benefit from progress in adjacent domains of algorithm research.

We emphasize that our choice of a 3D interconnect topology stems not from the 3D nature of intended applications but from the 3D nature of our physical universe. We anticipate that the NuMesh interconnect will perform well in any computation characterized by a static sparsely-connected communications graph amenable to efficient embedding in 3-space.

4.3.5 LegoFlops as a Research Goal

A major attraction of this scheme, and variants, is its promise of mind-boggling performance in a limited but conspicuous class of applications. Unlike more ambitious approaches to multiprocessor architecture, commitment of engineering talent, money, and corollary resources are almost certain to produce splashy results (i.e., huge MFlops and MFlops/parameters) and very impressive demos (speech, graphics, etc.). By riding the coattails of highly-engineered DSPs, we leverage the real muscle of the remaining domestic semiconductor industry; indeed, it is difficult to imagine TI and Motorola not vying strongly for an opportunity to participate (and to implant their respective DSP chips). The technical risks are low: the architectural schema can certainly be made to work; the software, while challenging at its most ambitious extreme, admits many clearly practical compromises. Perhaps the biggest challenges involve the basic physics of the system: mechanics, connectors, cooling, and power distribution. Here there is unlimited opportunity for cleverness, optimization, and state-of-the-art engineering.

Like most system-building proposals, this one is resource intensive: to achieve its potential, it will require staff engineers, VLSI fab, proselytizing and support of a user community, and the better part of a decade of LCS commitment. However, it seems to be an area where sufficient financial commitment virtually guarantees interesting and conspicuous results—results which, unlike the incestuous tools of the computer science community, enable breakthroughs in real (non-CS) applications. The ultimate attraction of this commitment to LCS and MIT may be the distinction it engenders in other areas: a period of MIT supremacy in speech recognition, imaging, dynamic graphics, control robotics, and a host of other client disciplines.

4.4 Alewife

The Alewife multiprocessor consists of a set of processing nodes interconnected via a low latency network. A high speed processor, a large coherent cache, memory and a cache-memory-network controller constitute each processing node. The current version of the network uses a topology of the Omega [198] (or Banyan) class of networks and is circuit-switched to allow low latency communications. Our research also addresses scalable fat-tree [203] and low dimension direct networks [91] that display locality and can provide quick access to neighboring memory modules without requiring a full network traversal. The processor, called APRIL [207], permits rapid context switching through the use of multiple register files and includes support for efficient synchronization and handling of Futures.

In the software arena, the parallel Mul-T system developed at the Laboratory for Computer Science in the Parallel Processing Group is being adapted for our use. The Mul-T system includes a production quality compiler for parallel applications. We have developed T-Mul-T, an address tracing system that produces traces of parallel applications written in Mul-T. T-Mul-T can also be interfaced to a cache-memory system simulator, which in turn interfaces to an interconnection network simulator. The coupling of trace generation and the memory system simulator allows rapid simulation of various system configurations without introducing time distortions in results; it also has speed advantages over a software processor simulator, and does not incur trace storage overhead. However, a processor simulator for our APRIL processor is also necessary, because APRIL is sufficiently different from the processor we are currently tracing. Such a simulator is currently being developed to replace the trace generation backend of our simulation system.

The principal areas of our research in the past year included:

1. Multiprocessor data collection tools and techniques;
2. The design of directory systems for large scale cache-coherent multiprocessors;
3. Low latency interconnection network design and analysis;
4. Investigation of new synchronization techniques;

5. Design of VLSI processors for parallel computers;
6. Parallel processing software and applications;
7. Exploiting locality to enable scaling of large scale multiprocessors; and
8. Performance modeling and evaluation.

The following is a brief description of each area.

4.4.1 Multiprocessor Data Collection

Continuing our efforts in parallel trace data collection, we now have a tracer called T-Mul-T that generates traces for parallel symbolic applications and is written under Mul-T, a parallel Lisp system described later. The first implementation was done for the Encore Multimax. The Mul-T kernel was modified by David Kranz to simulate an arbitrary number of virtual processors, running on only a single processor. The simulation switches to a different processor after each memory reference emitting a packet for each reference. The memory allocator was modified to make each processor allocate storage in its own area of memory so that we could study the effects of locality. The global memory allocation used by Mul-T would not make sense in a large scale multiprocessor. The context switching and memory packet emission is controlled by having the compiler insert code into the instruction stream. The resulting simulation is very fast, only 20 times slower than Mul-T on the Encore multiprocessor itself, neglecting I/O time for the memory packets if they are to be written out to disk.

T-Mul-T does two things for us:

1. It allows us to run a real Mul-T program on an arbitrary number of processors instead of just 16 (the number our Encore machine has). Because the simulation switches virtual processors after every memory reference, it gives a faithful simulation of a possible real execution of the Mul-T program, but it isolates scheduling and parallelism issues from the bus contention problems that exist in the Encore machine.
2. It gives us parallel traces for real programs that can be used in cache and memory network simulations to help understand the locality issues.

A port of T-Mul-T to the DEC Microvax and the MIPS R2000-based DECstation 3100 is also partially complete. We have gathered several large traces of symbolic applications written in Mul-T including MODSIM—a functional simulator, BOYER—a theorem prover, and several other applications.

In a joint effort with the IBM T. J. Watson Research Center, Mathews Cherian derived large parallel FORTRAN traces using a "postmortem scheduling method" that can incorporate multiple synchronization models. In this technique, a multiprocessor trace is created fro a memory reference trace of the uniprocessor execution of a parallel application. Using the record of synchronization events contained in the uniprocessor execution trace, a postprocessor can schedule tasks from the uniprocessor execution trace into a multiprocessor trace in which the synchronization sections are simulated assuming some model of synchronization. The scheduler simulates processors generating the requests in a round-robin fashion. Parallel FORTRAN traces of several popular benchmarks include SIMPLE, WEATHER, and FFT. We are using these traces in a wide variety of studies ourselves, and we plan to distribute our trace data to the research community and to industry. Efforts to trace these applications with modified algorithms to enhance program locality are also in progress.

We continued tracing parallel C applications under the MACH operating system using the Vax T bit technique [279][134]. Kiyoshi Kurihara has this tracer running on a DEC Microvax 3200 and is modifying it to use MACH threads instead of UNIX processes to enable faster tracing.

A slight modification to our parallel T-Mul-T tracer has also enabled the emulation of large scale multiprocessors, where the underlying processor on the machine which simulator runs on, substitutes for the processor in the multiprocessor being emulated. We have simulators for cache/directory systems and interconnection networks, which can be plugged back to back to provide the system backend to the processor emulator. The FORTRAN postmortem scheduler can also be used as the backend to the multiprocessor emulator.

4.4.2 Large Scale Cache Coherence

David Chaiken and Mathews Cherian worked on directory schemes and synchronization in large scale multiprocessors; earlier studies of directory schemes were limited to small scale systems of 4 to 16 processors. We investigated the scalability of limited directory schemes [10] for cache coherence in the large scale. In a limited directory scheme, the number of pointers in the memory for each block can be less than the number of caches in the system. Such a scheme works because of temporal locality property of processors referencing a given memory block.

Mathews Cherian wrote a cache and directory simulator suitable for simulating large scale systems that provide several statistics on cache coherence schemes such as the effects of cache size, block size, number of directory pointers, and the number of processors. The simulator also provides traffic rates of schemes that do not cache shared variables, and can distinguish between shared and private traffic, synchronization and non-synchronization traffic. On the slate for the future is obtaining statistics on a pointer-chaining directory based cache coherence scheme [76].

David Chaiken extended the work on directory schemes and wrote a simulator to reflect more details of the cache coherency protocol. This program can simulate a fully-acknowledged protocol that guarantees sequential consistency, correctly interacting with a rapid-context-switching processor. The simulator is being extended to handle a weak-coherence protocol that fields processor-issued fence instructions and outstanding memory operations, and to handle access to full/empty bit synchronization requests.

This program is intended to be part of a processor/cache/memory/network simulator that can be used to perform a detailed analysis of the architecture that we are proposing.

Results from simulations of parallel FORTRAN applications show that these limited directory schemes do scale for some applications. But for other applications, widely shared synchronization objects and sharing of words within cache blocks reduce their performance to almost that of a scheme that does not cache shared objects. Consequently, our current focus is on methods for restructuring parallel programs to exploit caches. Our results for applications written in Mul-T showed relatively much better performance due to the lack of widespread sharing prevalent in the FORTRAN applications.

A major observation was that synchronization references are another impediment to scalability. Because a large number of processors simultaneously access synchronization variables, excess traffic to a single hot-spot location results. Large scale, cache-coherent multiprocessors suffer significant amounts of invalidation traffic due to such synchronization reference patterns. Large multiprocessors that do not cache synchronization variables are often more severely impacted. If this synchronization traffic is not reduced or managed adequately, synchronization references can cause severe congestion in the network. We are investigating new scalable synchronization methods that do not incur excessive hardware cost [6]. A later section will describe our work that addresses this issue.

As a sampling of our results on the performance of directory schemes for cache coherence, Figure 4.4.2 shows an invalidation histogram for a 64-processor simulation of $Dir_N NB$ driven by a trace from the SIMPLE application. $Dir_N NB$ corresponds to a directory scheme that uses no broadcasts and has N pointers. The graph shows the histogram of the number of invalidations required during a write to a previously clean block. The graph shows the percentage of writes which resulted in invalidations to up to 12 caches. Writes resulting in invalidations of greater numbers of caches were proportionately insignificant. In over 95% of the times that an invalidation occurred, a block had to be invalidated from no more than three caches. This small number of invalidations compared to the possible maximum (i.e., 64) implies that for the common case the directory need have just a few pointers to encode the locations of the shared blocks. Invalidation histograms for FFT and WEATHER had a corresponding figure of over 99%. Synchronization references accounted for all the invalidations involving roughly 10 or more caches—a definite problem.

On a further investigation of the scalability of cache coherence schemes, we observed that with large block sizes, concurrent accesses of a block by several processors caused excess invalidation traffic. For example, halving the block size to 8 bytes from 16 almost halved the network request rate. Our current efforts are aimed at compiler techniques to make caches viable by reducing this interference effect.

David Chaiken is working on semantic models for shared memory. Ongoing research includes proving that our directory scheme implementation does conform to our definition of cache coherence [75]. The design of a cache-directory and network communications controller, to be used in a large scale multiprocessor, is in progress. The chief issues being addressed are: the programmability and the implementation efficiency of various shared-memory programming

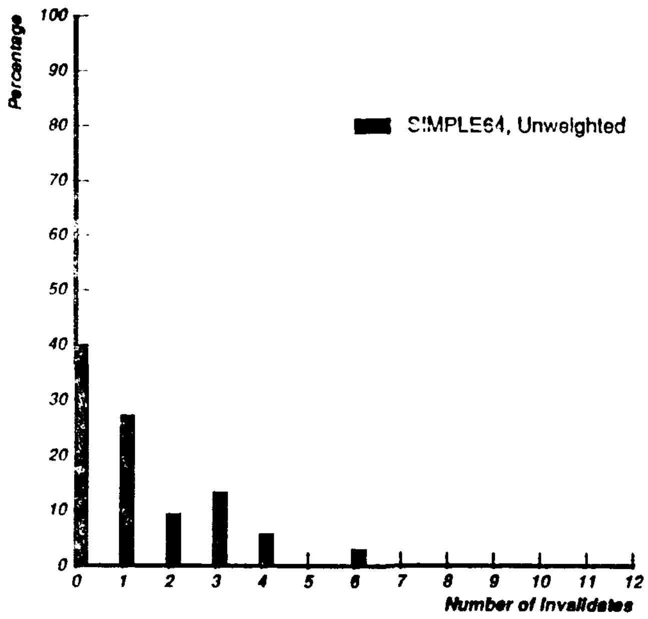

Figure 4.1: Cache invalidation statistics for SIMPLE with 64 processors. The height of a bar at x reflects the fraction of write hits to previously clean blocks that resulted in x invalidation messages.

paradigms, such as strong serialization versus weak ordering, supporting full-empty bits in the cache/memory controller, and tradeoffs in controller design to support context switching, such as re-issuing instructions versus pipeline freezing.

4.4.3 Interconnection Networks

We analyzed interconnection network architectures that can best exploit the lower average traffic intensity of cache-coherent systems. Analytical evaluations with packet-switched and circuit-switched networks, assuming similar speeds for the switch nodes, show that circuit-switching can be superior to packet-switching in the medium scale (256-1000 processors). Our simulations with the parallel FORTRAN traces also indicate that directories yield better processor utilization than a scheme that does not cache shared data. The relative advantage of caching can be further enhanced by clever program restructuring to exploit fast access to cached data.

Figure 4.4.3 shows the processor utilization for packet- and circuit-switched networks for a limited directory scheme with four pointers and a scheme that does not cache shared data for the FFT application. We see that circuit switching networks yield better performance up to about a thousand processors. Directory schemes are also superior to the non-caching schemes, although not by much. As mentioned earlier, we are investigating methods to improve benefits of large coherent caches.

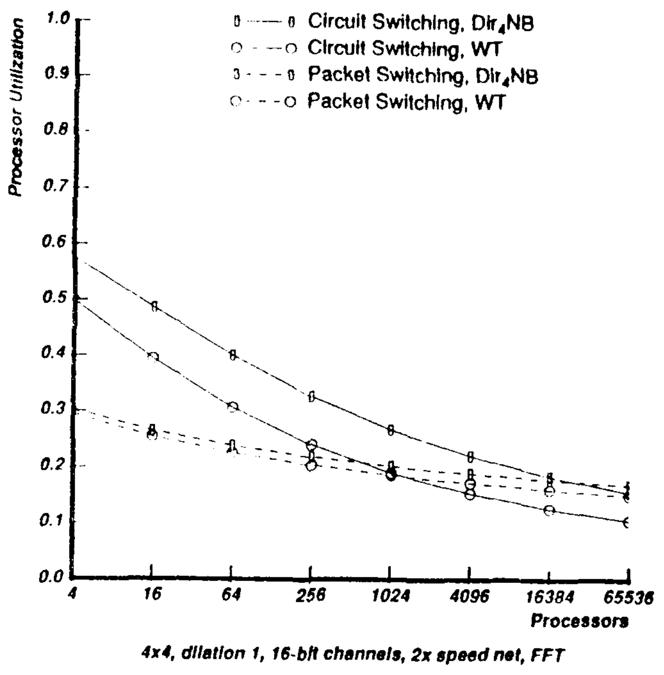

Figure 4.2: Processor utilization for packet and circuit switched networks for a limited directory scheme with four pointers and a scheme that does not cache shared data. The application is FFT.

Gino Maa has written a fully-configurable circuit-switched interconnection network simulator which is being used to model the performance of proposed machine architectures and to validate the effectiveness of our analytical performance models. It allows us to evaluate the impact of alternative processor architectures, cache and coherence protocol designs, and network topologies. A packet-switched version, being written by Sue Lee and Gino Maa will be operational presently, so that tradeoffs between circuit- and packet-switching can be examined via simulation under various system configurations. The simulator works with both live and static frontends: with live sources such as an instruction-set interpreter. block/resume handshaking signals are generated at the interface to exert back pressure on the execution and scheduling behavior of the processors. With static trace inputs, trace skew statistics is collected to provide a qualitative confidence measure on the integrity of the simulation data. In either mode, the network input traffic may be "pre-filtered" via a memory cache simulator. With the simulators, we are already collecting useful data which is guiding and confirming our design choices. One observation we made with our simulation thus far is that purely static backends that drive network simulations can be inaccurate. Measured maximum skews between the trace generated streams of various processors and those during network simulations were over a million references for a total simulation reference length of 20 million references!

A VLSI implementation of a circuit-switched network chip is in progress under the direction of Tom Knight [101]. Knight is also investigating high-density 3-D button-board packaging for the interconnection network.

4.4.4 Synchronization

We developed a new technique for efficient synchronization called *adaptive backoff synchronization* [6]. A purely software approach, adaptive backoff synchronization helps reduce network contention due to fine-grain synchronization accesses across a network. Our technique can also help reduce hot-spot contention in large scale networks without resorting to hardware-intensive solutions like combining networks [261] or global synchronization logic [164]. We are also investigating the application of these adaptive backoff schemes to reduce contention in source-responsible circuit-switched networks.

Our adaptive backoff methods use a synchronization state to reduce polling of synchronization variables. Our simulations show that when the number of processors participating in a barrier synchronization is small compared to the time of arrival of the processors, reductions of 20% to over 95% in synchronization traffic can be achieved at no extra cost. In other situations, adaptive backoff techniques result in a tradeoff between reduced network accesses and increased processor idle time.

We are also studying software combining [316] to determine the extent to which a directory cache coherence scheme can efficiently support fine-grain barrier synchronization. By using the postmortem scheduler for FORTRAN traces, along with some additional postprocessing software to simulate the effect of software barrier trees, Kiyoshi Kurihara is investigating methods to reduce synchronization costs in cache-coherent multiprocessors. The postprocessing program locates and changes spin-lock addresses to simulate a combining tree effect. Applications to both static and dynamically created barriers are being studied. To obtain results from the MACH tracing package, Kiyoshi is modifying the barrier macros to use combining trees and adaptive backoff methods.

4.4.5 Processor Design

We are investigating novel VLSI processor architectures for large scale multiprocessor systems. A processor called APRIL is being designed by Beng-Hong Lim and Dan Nussbaum [207]. This processor borrows heavily from the MARCH processor design by Bert Halstead and the Stanford MIPS-X processor [163], but differs substantially from the two. Unlike MARCH, APRIL has hardware interlocks in the pipeline, does not interleave process threads, and uses software thread scheduling. Unlike MIPS-X, it allows multiple hardware contexts, and has hardware support for synchronization and Futures. The chief issues being addressed in this design are rapid context switching, fast trap handling, high single thread performance, hardware support for synchronization and futures, and register file organization.

An important result of our study has been identifying the specific hardware-software tradeoffs for achieving overall high system performance. Some examples include hardware versus software for fine-grain task management and scheduling in a multithreaded processor, and hardware provided synchronization primitives such as fetch-and-op versus software synthesized primitives from basic interlocked load/store instructions. We currently have a preliminary instruction-set specification. A Mul-T compiler for this processor and a detailed simulator are also being written. Beng-Hong Lim has written an instruction-level simulator that recently ran the ubiquitous Fibonacci program.

4.4.6 Parallel Processing Software

David Kranz's work centered around Mul-T [189]. Mul-T is a parallel Lisp system, based on Multilisp's **future** construct [151], that was developed to run on an Encore Multimax multiprocessor. Mul-T is an extended version of the Yale T system [270][271] and uses the T system's ORBIT compiler [188] to achieve "production quality" performance on stock hardware—about 100 times faster than Multilisp. Mul-T shows that Futures can be implemented cheaply enough to be useful in a production-quality system. Mul-T is fully operational, including a user interface that supports managing groups of parallel tasks. People at other universities, labs, and companies are using Mul-T, and useful feedback is expected. (See [189] and the Parallel Processing Group report for more details.)

Mul-T is useful as a real system for parallel programming but suffers because it is difficult to do performance evaluation. We also do not want to limit ourselves to bus-based multiprocessors such as the Encore Multimax. For large scale multiprocessors it will be necessary to examine the effects of locality on performance. In order to get the data necessary to investigate these issues, David Kranz re-engineered Mul-T to get T-Mul-T described earlier T-Mul-T runs on an arbitrary number of processors independent of the number available in the host multiprocessor.

In collaboration with Susan Owicki, DEC Systems Research Laboratory, Palo Alto, we are investigating affinity-based process scheduling techniques for improving the locality of memory referencing in multiprocessors. This work uses analytical models of the performance of multiprogrammed single processor caches [8]. An analytical model of performance of multiprocessor caches has also been derived to be used in this study [251].

A continuing effort is the development of large parallel applications. A substantial benchmark program, SIMPLE, is being parallelized and ported to Mul-T, a parallel dialect of Lisp. It is a finite-difference numerical analysis program from the Lawrence Livermore Lab, which has become one of the standard benchmarks for evaluating existing and proposed high performance computers. The parallelization can be conditionally compiled to efficiently target a wide scale of multiprocessors (from 16 to 100's of processors.) Other programs already developed for Mul-T include parallel matrix multiply, Permute, and Modsim.

4.4.7 Multiprocessor Locality Studies

Caches can prove beneficial in large scale multiprocessor environments only if we can exploit locality in multiprocessor memory referencing to a much greater extent than we have been able thus far. Our measurement studies of parallel application traces confirm this need. Our efforts in this direction are summarized next.

Ongoing work aims at providing an integrated strategy to implement an efficient storage hierarchy in shared-memory multiprocessors. Currently, parallel traces and application programs are being used, along with the simulation tools, to analyze and characterize locality properties in the memory access traffic. This is preparatory work with the goal to explore ways to exploit memory access locality to reduce access latency and interconnection bandwidth requirements.

We have a new model representing memory referencing locality in multiprocessor systems [7]. This locality model suitable for multiprocessor cache evaluation is derived by viewing memory references as streams of processor identifiers directed at specific cache/memory blocks. This viewpoint differs from the traditional uniprocessor approach that uses streams of addresses to different blocks emanating from specific processors. Our view is based on the intuition that cache coherence traffic in multiprocessors is largely determined by the number of processors accessing a location, the frequency with which they access the location, and the sequence in which their accesses occur. The specific locations accessed by each processor, the time order of access to different locations, and the size of the working set play a smaller role in determining the cache coherence traffic, although they still influence intrinsic cache performance. Gino Maa has some initial results that show that these processor references directed to a memory block display the LRU stack property. If we succeed in showing this is indeed true across a large set of parallel applications, then the abundant literature on LRU stack evaluation for single processors can be straightforwardly used in evaluation of multiprocessor performance.

4.4.8 Multiprocessor Performance Modeling and Evaluation

Analytical models of computer performance become ever more important as we scale multiprocessors to hundreds or thousands of processors, where the computational needs of simulations far exceed those available to us now. In addition to the simulation and analytical modeling systems described in the previous sections, we developed the following performance evaluation models.

We developed a model of multiprocessor cache performance when coherence is enforced by the software [251]. A similar model for the performance of hardware-enforced cache coherence that takes into account the effects of the increase in invalidations as more processors are added is being developed in collaboration with Susan Owicki.

We have implemented several analytical models of network performance to predict effective processor utilization taking into account delays due to cache and network accesses and contention. The models are driven with access rates and access sizes measured from our benchmark traces. These models allow quick estimation of performance measures for various network configurations and numbers of processors.

Minor Huffman extended our trace compaction technique based on a model of the spatial locality in programs [5] to improve both the compaction rate and cache performance simulation accuracy [165].

4.5 Publications

[1] A. Agarwal. *Analysis of Cache Performance for Operating Systems and Multiprogramming.* Kluwer Academic Publishers, 1989. To appear.

[2] A. Agarwal and M. Cherian. Adaptive backoff synchronization techniques. In *Proceedings of the 16th Annual International Symposium on Computer Architecture*, IEEE, June 1989.

[3] A. Agarwal and A. Gupta. Memory-reference characteristics of multiprocessor applications under MACH. In *Proceedings cf ACM SIGMETRICS 1988*, May 1988.

[4] A. Agarwal and A. Gupta. *Temporal, Processor, and Spatial Locality in Multiprocessor Memory References.* MIT VLSI Memo, April 1989. Submitted for publication.

[5] A. Agarwal, J. Hennessy, and M. Horowitz. Cache performance of operating systems and multiprogramming. *ACM Transactions on Computer Systems*, 6(4):393–431, November 1988.

[6] A. Agarwal, M. Horowitz, and J. Hennessy. An analytical cache model. *ACM Transactions on Computer Systems*, May 1989.

[7] A. Agarwal, R. Simoni, J. Hennessy, and M. Horowitz. An evaluation of directory schemes for cache coherence. In *Proceedings of the 15th International Symposium on Computer Architecture*, IEEE, June 1988.

[8] S. Owicki and A. Agarwal. Evaluating the performance of software cache coherence. In *Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS III)*, April 1989.

[9] R. L. Sites and A. Agarwal. Multiprocessor Cache Analysis using ATUM. In *Proceedings of the 15th International Symposium on Computer Architecture*, pages 186–195, IEEE, June 1988.

[10] S. Ward and R. Zak. Static-column RAM as virtual cache. *Proceedings of the Eighth International Conference on Computer Science*, Santiago, Chile, July 1988.

[11] S. Ward and R. Zak. Set associative dynamic random access memory. *Proceedings of the International Conference on Computer Design*, Port Chester, NY, October 1988.

Theses Completed

[1] E. Anderson. *A Million-pixel High Performance Graphics Card.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

[2] T. Bloomstein. *Audio and Control Data Switching Station.* Bachelor's and Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

[3] M. Cherian. *A Study of Backoff Barrier Synchronization in Shared-memory Multiprocessors.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[4] B. Dennis. *An Error Handler for L.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

[5] J. Elsbree. *Declarative Graphical Programming for Automated Circuit Test.* Bachelor's and Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

[6] M. Huffman. *A Spatial Locality Based Trace Compaction Method.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, April 1989.

[7] D. Jedlinsky. *A PostScript Interpreter for the Apple Macintosh.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

[8] S. Kommrusch. *Microarchitecture Design and Simulation for an L Processor.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

[9] J. Nguyen. *A Type Inference Algorithm for the Language L.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

[10] P. Pilotte. *A Virtual Digital Multimeter for the Macintosh.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

[11] M. Roberts. *The DAANub: Digital Audio Access for the NuBus.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

[12] T. Steele. *Dream II, Dynamically Reconfigurable, Electrically Alterable Module II.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

[13] J. Wang. *HyperCommunicative User-friendly Telecommunications.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

Theses in Progress

[1] T. King. *Test of Metastability in Synchronizer Circuits.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected August 1989.

[2] J. Morrison. *Scalable Multiprocessor Architecture Using Cartesian Network-relative Addressing.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected August 1989.

[3] M. Powell. *A Simulator for Multiprocessor Implementations of L.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected June 1989.

[4] E. Puckett. *A Machine-independent Intermediate Language.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

[5] N. Osgood. *Automatic and Near-exhaustive Derivation of Machine-dependent Optimizations.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, June 1990.

Talks

[1] A. Agarwal. High-performance computer design. Lecture given at Bell Communications Research, NJ, July 1987.

[2] A. Agarwal. Cache performance of operating systems and multiprogramming. Lecture given at Digital Equipment Corporation, Hudson, MA, June 1987.

[3] A. Agarwal. Can we build large-scale shared-memory multiprocessors? Lecture given at IBM T. J. Watson Research Center, Yorktown, NY, March 1988.

[4] A. Agarwal. And now, the right way to build large-scale shared-memory multiprocessors. Lecture given at MIT Laboratory for Computer Science, March 1988.

[5] S. Ward. L: a computer architecture for the coming decades. Lecture given at Schlumberger, Austin, TX, August 1988.

[6] S. Ward. The L project. Lecture given at Apple Computer, Cupertino, CA, September 1988.

[7] S. Ward. The L project. Lecture given at NeXT, Inc., Palo Alto, CA, November 1988.

[8] S. Ward. Computer systems architecture: a local perspective. ILO presentation to Intel, MIT, April 1989.

Computation Structures

Academic Staff

Arvind, Group Leader

J.B. Dennis	R.S. Nikhil

Research Staff

G.A. Boughton	G.M. Papadopoulos
J. Young	R.P. Johnson

Technical Staff

J.P. Costanza	R.F. Tiberio

Graduate Students

P.S. Barth	S.K. Heller	V. Kathail
S.A. Brobst	M. Heytens	B.C. Kuszmaul
D.E. Culler	J.E. Hicks	J.S. Onanian
B. Guha Roy	A. Iyengar	S. Sharma
S.A. Gupta	S. Jagannathan	R.M. Soley
D. Henry	C.F. Joerg	K. Steele

Undergraduate Students

D. Chiou	I. Scharf
Y. Chery	A. Shaw
C. Fabian	D. Stetson
S. Furman	P. Tan
L. Muryanto	G. Wang
J. Santoro	

Support Staff

S.M. Hardy	N.F. Tarbet

Visitors and Adjunct Members

A. Altman	S. Landsberg
Z. Ariola	M. Sadoune

5.1 Introduction and Overview

Our group is interested in general purpose parallel computation. Our approach is centered on:

- Declarative, implicitly parallel languages.
- Dataflow architectures, which are scalable because of their tolerance of increased memory latencies and support for frequent synchronization. Our vehicles for research include an abstract "Explicit Token Store" architecture (ETS), a hardware prototype implementation of ETS (Monsoon), various software emulators (Gita, MINT), a new proposed architecture called P-RISC, and a software emulator for it.
- Sophisticated compiling and runtime systems for Id, both for dataflow and other architectures. We have also explored the use of dataflow compiling for an experimental persistent programming language to tolerate disk latencies by exploiting parallelism.
- Applications programs to guide the language, compiler, and architecture research.

Our main research vehicle for programming languages is Id, which is a mostly functional programming language. We completed the basic type system and are exploring the use of a simplified version of a new overloading mechanism due to Phil Wadler [306]. Id is a nonstrict language for more parallelism, but nonstrictness is not achieved via laziness, as is usually the case. Instead, we have explored the implications of using explicit constructs for lazy evaluation to deal with infinite structures. For nondeterministic access to shared state, we have developed a new construct called a "manager" that is similar to, but more flexible than monitors and also allows more concurrency. We have also explored a few other experimental language designs: a language with naming environments as first class objects, and a language for signal processing. Our group is well represented in the international committee that is designing the new functional programming language Haskell.

On the more theoretical side, we have formalized Id's operational semantics using rewrite rules, and have been able to prove results about determinacy and to be more precise about such concepts as termination, errors, etc. We have also studied optimal interpreters for the lambda-calculus.

We have ported a subset of Id World, our programming environment for Id, to the UNIX environment. This should make Id available to a much larger audience. The UNIX version lacks the graphics of the original Lisp Machine version; this work remains to be done.

Last year, we reported that our research results had reached a level of maturity where we were ready to embark on the construction of a real dataflow machine within the next few years using the Monsoon processor architecture (Project Dataflow). Towards that end, we held a meeting in March 1988 with prospective industrial partners. Over the last year, Motorola has emerged as our partner; they are setting up a Cambridge research laboratory, and will participate actively in the construction of the Monsoon system.

A wire-wrap prototype of a processor using the Monsoon dataflow architecture has been running small handcoded programs since September 1988, and has compiled code since December. It has been used to guide the design of the printed-circuit Monsoon board (part of Project Dataflow). We continued to make progress on the design and implementation of the Monsoon interconnection network, consisting of PaRC switching chips and high speed data links. We have begun work on the design of an I-structure memory board for Monsoon.

We have incorporated more optimizations in the Id compiler, and are moving its target away from the Tagged Token Dataflow Architecture to an Explicit Token Store model (ETS), of which Monsoon can be considered a specific implementation. We began to look very seriously at the runtime system and the control of parallelism in Id programs for better resource management, and have implemented several experimental mechanisms to that end.

Our repertoire of Id applications continues to grow and includes DNA sequence analysis, airport landing approach planning, computational fluid dynamics, image processing, and simulated annealing.

Our architecture research has also moved further in the direction of achieving a synthesis between von Neumann and dataflow ideas. We proposed a new architecture called P-RISC (for "Parallel RISC"), and have begun simulation and compilation studies.

Based on the I-structure notation in Id, we have designed a "functional database language," in which data do not change—update transactions specify new versions of a database. We are implementing this database language, using ideas from P-RISC compilation to exploit parallelism to hide disk latencies.

5.2 Personnel

After finishing his Ph.D. thesis in August 1988, Greg Papadopoulos became a member of research staff, working as the chief architect for the Monsoon prototype processor in Project Dataflow.

After completing his Ph.D. with Paul Hudak at Yale, Jonathan Young joined us as a member of the research staff, working on the compiler backend and runtime system for Monsoon. His research is in compile-time semantic analysis and optimization of functional programs.

Paul Johnson joined our research staff and has been working on porting the existing Id World to UNIX machines.

Arthur Altman joined CSG in January 1989 as a visiting researcher from Texas Instruments, to study the dataflow approach to programming languages and architectures when applied to problems in image understanding.

After finishing his Ph.D. in May 1988, Ken Traub stayed on as a research staff member. In early 1989, he joined the Cambridge Research Center of Motorola, Inc., the industrial partner on the Monsoon project.

It is with great sadness that we record the passing of Bhaskar Guha Roy on March 23, 1989. He worked first with Jack Dennis and later with Prof. Nikhil. He fought an incredibly courageous, year-long battle against liver cancer, during which he managed to write his Ph.D. thesis proposal and set up his committee.

5.3 Programming Languages

5.3.1 Id

In September, we released the reference manual for Version 88.1 of the Id programming language [244], which augmented the language with constructs for loop bounding.

5.3.2 Types and Overloading

During the Summer and fall of 1988, Shail Aditya revised and upgraded the type checking system of the Id compiler to incorporate changes from Id'87 to Id'88. This involved the addition of several key features to type analysis, viz., algebraic data types, constructor case analysis and abstract data types. Further, the type checker was made totally incremental at the procedural level. Thus, in the version currently installed, the user can compile individual procedures interactively from the editor, in any order. The type checker, installed as a module in the Id compiler, incrementally assembles enough information to check the type consistency of the accumulated program at each interactive step. Using this information, the runtime environment is able to double check the type consistency of all the procedures in the invocation graph just before execution. The user is notified in case of any discrepancy and the appropriate section of the program can be corrected and recompiled.

During the winter and spring of 1989, Shail Aditya worked on a mechanism for the resolution and compilation of overloaded of operators and general user-defined identifiers. The idea is a simplification of the system proposed by Wadler and Blott [306] which has been adopted in Haskell. Unlike previous overloading schemes, this one is not ad hoc. It is capable of expressing "recursive overloading", e.g., if "+" is already overloaded on integers and floats, then it can also be overloaded to mean addition of lists of integers and floats and, inductively, on lists of lists of integers and floats, etc. There is a systematic way of resolving this overloading.

The type checker with overloading resolution is currently under test with regards to efficiency of compilation and execution. We are conducting experimental tests with existing Id programs including large scientific codes such as SIMPLE. It will be installed in the Id compiler in the near future. The proof of consistency of the incremental type system and the details of the overloading mechanism are due to appear in Shail's forthcoming S.M. thesis.

The straightforward resolution of overloading results in some inefficiency because a procedure that uses the symbol "+" is implemented as one that receives an addition function as a parameter, which is applied using a general function call. It remains to be seen how this can be optimized through a process called "specialization", where separate versions of the procedure are compiled, one for each implementation of "+" that is of interest.

5.3.3 Lazy Evaluation

Id has nonstrict semantics, which means that a procedure or data constructor application can produce a value before the value of its arguments are known. Traditionally, languages with nonstrict semantics have been implemented using lazy evaluation, where nothing is evaluated until it is known that it is needed for the result. Unfortunately, when an expression *is* needed, a lazy evaluator would have already paid the overhead of building a closure for the expression and rescheduling it. Further, it would have lost the opportunity of evaluating it concurrently with other computations. For these reasons, we choose not to use lazy evaluation in Id.

However, lazy evaluation can be very useful for programming with infinite structures (e.g., streams), and for large data structures of which only a small part is actually used. Steve Heller completed his Ph.D. thesis in January 1989, in which he investigated the design, use and implementation of explicitly designated lazy data structures in Id [158]. Heller and Jamey Hicks implemented lazy data structures in the graph interpreter (Gita) based on some preliminary work of UROP student Chuck Fabian. He was able to show that of the numerous examples of applications that used lazy evaluation in the literature, most of them needed only nonstrictness, not laziness. The few instances where laziness was actually necessary were easy to identify, and it was quite easy to use the explicit lazy data structures in Id. Jonathan Young and Hicks implemented a restricted version of lazy data structures on Monsoon (four states instead of five states in the state diagram, since Monsoon only has two status bits). Lazy data structures are being used to implement global constants and for stream programming, and have also been used in system code for memory allocation.

5.3.4 Managers

Paul Barth continued his research on *managers*, a construct for supporting nondeterministic computation in Id. Nondeterministic constructs are needed for state-sensitive computation, including "application" programs, such as real time systems and database systems that respond to multiple inputs according to their temporal order. They are also necessary for "systems" programs, such as runtime support for the implementation of a functional language, which need to manipulate the state of the machine.

The manager construct was redesigned to facilitate programming abstraction and efficient implementation. Rather than stream functions, managers have been recast as abstract data types, with operators that access and update a shared state. This is beneficial from two standpoints. As a programming construct, this makes the nondeterminism explicit while encapsulating the state transformation. Each potentially nondeterministic operator is easily identified, and can be written as a function from old state to new state.

Managers are similar to monitors, but allow much more flexibility in scheduling the queues of waiting processes, and allow much more concurrency between state-manipulating procedures.

From an efficiency point of view, the new paradigm allows mutual exclusion to be provided by hardware primitives rather than stream operations. These primitives, called *locks*, are an extension of I-structure operations that provide efficient mutual exclusion on individual memory cells. The design of locks (developed jointly by Barth, Soley, and Steele) is currently being filed for patent. The new manager construct is fully described in CSG Memo 294.

Managers were incorporated into the compiler, and applications were developed, including the dining philosophers problem, a shared bank account (with deferred debits), a printer scheduler, a buddy system memory allocator, and a union-find set algorithm. These examples indicated that the new design was more perspicuous and efficient than stream-based managers.

5.3.5 Other Language-related Work

Sequential Implementations of Nonstrictness

Ken Traub's work on sequential implementation of nonstrict programming languages has continued, resulting in a paper presented at the Aspenäs Workshop on the Implementation of Lazy Functional Languages in Göteborg, Sweden. The paper is also to be presented at the 1989 Conference on Functional Programming Languages and Computer Architecture in London.

Symmetric Lisp

Suresh Jagannathan completed his Ph.D. thesis [169] on Symmetric Lisp, a novel parallel programming language in which naming environments (called *maps*) are first class objects. Through numerous programming examples, he was able to show that many diverse programming paradigms and constructs from other languages can be expressed quite elegantly with just the map construct. Examples include records, LET and LETREC blocks, "object-oriented" programs, file systems and directories, etc.

Using a single construct (the map), both as a data structure as well as a control structure, raises some interesting questions about formal properties of programs, because names are used both as program variables and as field selectors. For example, in the expression:

```
(with M e)
```

a free name `x` in `e` is looked up in `M`, if `M` is a map with a field `x`; otherwise, it is looked up in the surrounding lexical environment. Jagannathan developed an inference algorithm to produce statically a conservative approximation that predicted which environment a name would be looked up in. A compiler could use this information for efficient compiling name lookup efficiently. He also showed an implementation of Symmetric Lisp in terms of a translation to dataflow graphs.

Optimal Interpreters for Lambda Calculus

Vinod Kathail continued his investigation of optimal interpreters for the λ-calculus and functional languages based on the λ-calculus. The work in the last year focused on two aspects of the interpreter we had developed: formally proving its correctness and optimality, and simplifying its exposition. To relate our interpreter to the λ-calculus, we developed a new term calculus, which captures some of the essential features of the way the substitution operation of the λ-calculus is implemented in our interpreter. The term calculus is used as an intermediate step in proving the correctness of our interpreter; however, it may be of interest in its own right. We are in the process of completing the formal proofs [175].

PGL, A Signal Processing Language

Janice Onanian completed a Master's thesis in spring 1989, in which she developed a high level, signal processing language, called PGL, and a program graph representation for coarse-grain multiprocessors. Effective use of parallel processors requires dividing an application into concurrently executable tasks and assigning those tasks to processors such that their use of the network resources is optimized. We plan to use the language and graph developed in the thesis to find an optimal partitioning of an application into parallel tasks for a given hardware configuration. This involves two efforts: the development of algorithms for evaluating a task partition denoted by the program graph; and finding the optimal partition by varying the parameters to the program graph. Implementation of the PGL compiler is targeted for summer 1989; and development of the evaluation and optimization algorithms is planned to form the basis for subsequent, doctoral research.

Haskell, A New Functional Programming Language

Arvind and Nikhil have continued to participate in the design of the new functional programming language, Haskell. As reported last year, Haskell is being designed by a group of about 20 functional programming researchers from three continents. A draft of the language report was released to the public for comments in December 1988, which was followed by extensive discussion on the FP (functional programming) mailing list. The Haskell committee then met again in Mystic, CT in May 1989, where we charted the design decisions and actions to be taken before the final report is released in July 1989.

5.4 Id World: The Id Programming Environment

During the fall, R. Paul Johnson implemented a suite of interface functions designed by Richard Soley for Gita, the graph interpreter. This suite of functions, known as the Id World Interface (IWI) will support a variety of Id World user interfaces. Id World Version 4.0 and earlier only provided a Lisp Machine-specific graphical interface. Id World Version 4.'

includes a portable Common Lisp-based command listener. An X Window-based interface is under development. With the assistance of Jamey Hicks, Johnson released version 4.0 for internal testing in early December. Version 4.0, with support for Symbolics Genera 7.1/7.2 and TI Explorer 3.2/4.1, was shipped in January. Highlights of Version 4.0 include an optimizing compiler for Id 88.1, Id Mode Zmacs editor support, and the Gita graph interpreter with support for top level constants. Version 4.1, which adds support for Lucid Common Lisp Version 3 on Sun Workstations, was released externally for beta test in late March.

The next version of Id World will have greater separation between modules than in the current version, so that each piece may be run separately in a UNIX environment as opposed to being tied to the Lisp Machine implementation. In addition, Hicks has been meticulously documenting the internals of the runtime managers and the compiler schemata used in the current system, as well as some of the desirable hacks on the new hardware.

5.5 Project Dataflow: The Monsoon Prototype System

5.5.1 The Monsoon Processing Element

A very exciting milestone was met in September of 1988 when a single processor Monsoon prototype was made operational, able to execute incrementally compiled Id88 programs. The prototype implementation was engineered by Jack Costanza and Ralph Tiberio in compliance with the Monsoon microarchitecture specification developed by Greg Papadopoulos [253].

The Monsoon prototype is a 64-bit, fully pipelined (eight stages) dataflow processor. Constructed from off-the-shelf components on a single large wire wrap panel (9U $\times$ 600mm), the processor processes a modest four million tokens per second or approximately three dataflow MIPS of which any proportion can be double precision floating point. The processor board is enclosed in a custom cabinet with suitable power supply and cooling, and then connected via ribbon cables to a simple NuBus interface card hosted in a Texas Instruments Explorer Lisp Machine.

Hardware verification and debugging was facilitated by two design disciplines. First, we performed thorough timing simulations of entire board on our Mentor design tools. During simulation we executed small dataflow graphs to verify overall operation and focused specifically on various matching operations token enqueuing sequences. The second design discipline was to employ *scan paths* for (almost) all internal state. In scan path design, each parallel register can have its contents read and written through a special serial path, and multiples of such registers have their serial paths concatenated and then looped back to form a large *scan ring*. Any bit of processor state can be accessed by shifting these serial registers. Finally, the scan rings can be read and written through NuBus operations performed by the host Lisp Machine.

The prototype processor comprises over 800 bits of scannable state. Software on the host Lisp Machine interprets and displays the processor state in a full screen format, with appropriate data conversions (e.g., floating point) and mnemonics (e.g., opcodes, field decodings). The prototype processor clock can also be single stepped under host control, and by repeatedly stepping the clock and scanning state a full suite of software breakpoint conditions can be established. In essence, we used the combination of scan path design and host software to develop a sophisticated in-system logic analyzer. We found this to be a very effective debugging technique.

The Monsoon prototype forms the basis for the production Monsoon processor, a printed circuit board version to be manufactured by Motorola. Several improvements are incorporated in the production version.

- A network port based on the PaRC and link chips is added to permit the construction of multiple processor systems.
- A set of exception mechanisms and more complete support for system programs (e.g., loader, garbage collector) have been designed.
- The host interface has been changed from NuBus to VME and a high bandwidth DMA path has been added from the host into Monsoon frame store.
- The instruction format has been changed slightly to permit a wider opcode field (from 10 bits present to 12 bits) and variant formats are introduced that allow either two explicit destinations or a large absolute address displacement (20 bits).
- Much of the datapath has been byte sliced into 10,000 gate CMOS arrays (eight identical slices) and the specialized ALU functions that manipulates tags (the Pointer Increment Unit) has been cast by George Wang into a similar sized array.
- The pipeline rate has increased to ten million tokens per second, approximately seven million dataflow instructions per second.
- The board size has been reduced from 9U $\times$ 600mm to 9U $\times$ 400mm ("Sun size") through the use of gate arrays and surface mount assembly.

The production processor is in the final detailed design and simulation phase. We expect to hand off the design to Motorola by June 1989.

5.5.2 The Interconnection Network for Monsoon

Andy Boughton, Chris Joerg, and John Santoro continued their work on the network for Monsoon. We have continued to develop the two chips that will be used in the network, the Packet Routing Chip (PaRC) and the Data Link Chip (DLC). PaRC is a four input four output packet router on a chip and is the primary component of the Monsoon network. DLC contains a data link transmitter and a data link receiver. The transmitter will allow a PaRC output port to be connected to an interboard cable and the receiver will allow an interboard cable to be connected to a PaRC input port.

Joerg has continued the development of PaRC; the design has not changed significantly over the past year. Some work has been done to enhance the statistic collection abilities. Also, some improvements were made to the control port of PaRC. The control port is the section that allows a local controller to control several parameters of the chip's operation (such as how to do routing and what to do when errors are seen). Most of the work done on PaRC has involved creating test vectors. These vectors will be used to ensure that fabricated chips do not contain any functional defects.

Santoro continued the development of DLC. During the past year, we have used the preliminary logic design completed last year to develop a detailed design for DLC in Motorola's Mosaic II ECL gate array technology.

The top level design of DLC has changed somewhat during the year. The primary change was the elimination of 4 into 6 encoding. Our original design called for the encoding of all data transmitted over interboard cables. The primary advantage of this encoding was the elimination of the DC component of the transmitted signal. However, encoding required that the DLC be designed to operate on a 50% faster clock. Designing DLC for such a clock turned out to be a fairly difficult task. Faced with this difficulty, we ran a large number of tests on our proposed drivers, receivers, and cable to determine whether data could be reliably transmitted without encoding. Our tests indicated that a data pattern containing an arbitrarily long sequence of 0's followed by a 1 and another long sequence of 0's could be transmitted over the cable with more than sufficient noise immunity. Our tests indicated that the inverse pattern also worked. Based on these tests we elected to simplify design of the DLC by removing encoding.

The detailed design of DLC has been completed and simulated. Test vectors have been written which are sufficient for testing fabricated chips for faults. A preliminary version of the design has been transferred to Motorola. The final version of the design should be given to Motorola before June 30, 1989.

5.5.3 The I-structure Memory Board

During the spring, Ken Steele began work on a hardware I-structure controller design that will implement I-structures and the new memory operations developed for the Monsoon prototype. Each board is expected to provide 4MW (64 bits/word) per board, and be capable of handling up to five million requests per second through an onboard PaRC chip.

5.5.4 MINT: a Monsoon Simulator

Andy Shaw and Jonathan Young implemented a simulator for the Monsoon architecture which proved to be an invaluable tool for debugging the hardware. For his S.B. thesis, Shaw then extended this project into a complete interpreter that is capable of mimicking the hardware with great precision. The intent is that any object code that runs on Monsoon will run without modification on MINT. The design is very modular, and uses the Monsoon microcode compiler described below.

Since we wish to accurately simulate the processor, regardless of its current microcode, a microcode to Common Lisp compiler was designed and coded by Derek Chiou. The compiler accepts Monsoon microcode and translates it into Common Lisp comparable to hand code in efficiency. Secondary considerations were human readability and code size. Thus, the identical microcode specification used to drive the actual hardware is also compiled for the simulator, with obvious benefits of hardware – simulator consistency. The compiler is flexible enough to adapt to any foreseeable microcode changes. The compiler is written in Common Lisp.

5.6 Implementations of Id

5.6.1 Id on Monsoon

Jonathan Young and Jamey Hicks spent most of their time this year porting the existing Id compiler to Monsoon (with some initial work by Bradley Kuszmaul). This enables us to run real programs on the Monsoon wire-wrap prototype. We now have a working Monsoon compiler, as well as a loader, a runtime system, an execution manager, and a rudimentary debugger; the standard libraries have also been ported.

While much of the work of porting the compiler was easy because the Monsoon ETS architecture strongly resembles the previous TTDA architecture, the runtime system required major work. The Gita simulator relied on the storage management of the Lisp Machines; on Monsoon we implemented handcoded managers for free lists, frames for procedure calls, and two different heaps. We also implemented managers for I-structures and semaphores ("locks") to tide us over until we have a working I-structure memory board. In addition, special managers were needed to support particular language features such as delays and accumulators.

Using the execution manager, the user may now call any Id function which has been compiled and loaded into Monsoon with as many arguments as desired. Execution is currently limited to either "run until done" or a general single stepper, in which all eight stages of the Monsoon processor pipeline are visible. After a program error has been detected, various tools allow the user to view waiting tokens, data structures, and instruction memory.

Barth and Young developed a graph browser, and Doug Stetson improved the display heuristics. The browser proved to be very useful for debugging both the compiler and Id programs, and it has been installed in the Monsoon system. When compiling, the graph of a procedure is optionally displayed; it is also possible to view the graph of a procedure with its waiting tokens after a partial execution on Monsoon.

5.6.2 Storage Management

Ken Steele wrote microcode for the processor prototype to simulate I-structures in the processor's memory. Also, new instructions were created to support the runtime environment and compiler. These included non-busy waiting locks and support for lazy evaluation. A patent has been applied for on the non-busy waiting locking mechanism.

A storage management system was implemented in Id for dynamic allocation and deallocation of structure memory and frame memory. Stephen Brobst implemented a buddy system algorithm using nondeterministic lock and unlock primitives, which were provided as extensions of the Id language as a result of work done by Barth. Multiple instantiations of the allocation and deallocation routines can proceed in parallel, with suspension occurring only when two allocations attempt to allocate blocks of the same size. Fast path execution of the memory allocation routine requires less than 50 RISC-like instructions for its critical path. Young ported the buddy system to the Monsoon Architecture and augmented the storage management system with stack-based allocation mechanisms for cons cell and fixed-size frame memory allocation. Brobst has also written an Id version of the first-fit algorithm and is experimenting with various granularities for free list management.

A first version of the Id runtime system has been specified and is now under implementation. The storage management system will leverage the work of Barth, Brobst, and Young to provide dynamic allocation of structure storage and frames for large codeblocks using the buddy system, and in-line stack allocation of memory for cons cells and fixed-size frames. The I/O subsystem will provide a primitive interface to the file system using string objects and standard system call interfaces for file open, close, read, and write. Extensions to the Id language for synchronizing multiple reads and writes to a single file are an active area of research.

5.6.3 Long Term Software Structure

The above retargeting of Id for Monsoon uses the TTDA code that is produced from the existing backend of the compiler. This is not an attractive route in the long term. Young has written a specification of the ETS abstract machine [317] for use in compiling to the Monsoon architecture as it slowly evolves.

Traub has designed the architecture of the software system which will support Monsoon, to be jointly implemented at MIT and at Motorola Cambridge. The greatest difference between the new software system and the old TTDA/Gita system is one of modularity. Whereas the functions of loading Id programs, running them, debugging them, and displaying runtime statistics were previously all handled by the Gita program, in the architecture each of these functions will be handled by separate programs, with a top level program provided to present the user with essentially the same programming environment as found in the current Id World. The resulting system will be much more robust and flexible, and will point the way for the eventual migration of these functions onto the dataflow processor itself. Perhaps even more importantly, these programs are designed to work both with Monsoon hardware and its software emulation (MINT). Local area networks are an important part ot the new system, both in the use of X Windows as the framework for the user interface and in providing a network path to the Monsoon hardware or emulator. This will allow for easy sharing of a Monsoon system among several users. The software architecture and all the interfaces between its components are thoroughly documented in [301], edited by Traub.

Hicks and Traub designed the Monsoon Object Code (MOC) format. This is the format in which the Id compiler (and other programs) will write object files. MOC is based on CIOBL (Common Input/Output Base Language), redesigned by Traub.

Hicks had also made an initial design of the Id Object Format. The Id Object Format describes the data structures that will be loaded for an Id program. There will be structures for each procedure, global constant, and code block compiled and loaded. These structures will hold computed values and program code, and will support dynamic linking. They will also have source information for use in debugging. All program information that is needed at runtime will be structured using the Id Object Format. The actual object files will be encoded into MOC.

5.6.4 Experiments with Structure-storage Management

Jamey Hicks extended the Id compiler to handle data structure release annotations. This allows us to deallocate data structures relatively painlessly, but it is not meant to be a language feature that users will employ. It is meant to be an experimental feature, so that we can compare the performance of hand-annotated programs with that of automatically-annotated programs.

The syntax of the annotation is:

```
@release IDENTIFIER;
```

or

```
@release IDENTIFIER_0, IDENTIFIER_1, ...  IDENTIFIER_n;
```

inside a block expression. This annotation specifies the release of the structure bound to IDENTIFIER , when all computation enclosed within the block expression has terminated. This only releases the storage corresponding to the top level of the structure; the compiler cannot determine how much sharing of substructures there are in the program, so it does not release them. The compiler inserts the synchronization code necessary to ensure that the object is not released until all of the code in the block has terminated computation.

Inside a loop, @RELEASE actually has two meanings: if the structure is not circulated, then it is released when the current iteration has terminated; otherwise, if the structure is circulated in the loop, then it releases all but the first and last values of the structure when the corresponding iteration has terminated. The release of circulating structures is accomplished by unrolling the loop once, and not releasing the structure in the initial execution of the body.

Here is an example of the `@RELEASE` annotation in the `multiwave` procedure:

```
%%% Run several iterations of the wavefront.
%%% Illustrates the arbitrary chaining achievable in dataflow.
%%% ***** Release the intermediate waves when through with them.
def multiwave edge_vector n =
  {m = initial_wave edge_vector;
   in
     {for i <- 1 to n do
        next m = wave m;
        @release m;
      finally m }};
```

Young has written a simple compile-time analysis program which determines when it is safe to deallocate structures in loops; deallocation annotations are then automatically added to the program.

5.6.5 Resource Management in Scientific Programs

David Culler made substantial progress this year toward effective management of parallelism and resources in dataflow programs. The problem is that exploiting parallelism to achieve high performance invariably increases the resource requirements of a program. This phenomenon is not particular to dataflow, it can be observed to some degree in any form of parallel execution. However, it is particularly serious under dynamic dataflow execution, because all the potential parallelism in a program is exposed. This means that ample parallelism is available on a broad class of programs but, unfortunately, the resource requirements of many programs are excessive, often leading to deadlock. Culler documented both sides of this dilemma using parallelism and resource profiles of a variety of scientific programs derived under an ideal dataflow execution model (supported by Gita).

In 1985, he developed a mechanism for controlling parallelism, called k-bounded loops. Basically, loops are compiled into dataflow graphs in a manner that allows the maximum number of concurrent iterations to be set dynamically, when the loop is invoked. This approach is appealing for scientific programs, which are dominated by iterative computations over large, regular data structures. It has played a central role in the evolution of tagged token dataflow architectures toward Explicit Token-Store machines and hybrid machines because it allows the tag space to be used densely. Also, it provides a natural means of reusing resources within iterative computations. The question he has been exploring recently is how to assign the k bounds automatically.

The approach Culler has taken is to rely heavily on static analysis to characterize the dynamic behavior of programs. There are two aspects of this analysis: worst case resource requirements and expected parallelism. A representation of the dynamic call structure of the program is constructed and annotated with symbolic *resource expressions* which are parametric in the k bounds and in certain program variables. In addition, loops are classified as having limited useful unfolding, expensive unfolding, and efficient unfolding. Based on this analysis, the program is augmented with resource management code that computes the k-bounds by simple formulae derived from the resource expressions that capture a high level policy, e.g., favor the middle level in this triply-nested loop. A variety of policies have been examined analytically and empirically, and a particular policy has been effective in containing the resource requirements of scientific dataflow programs, while exposing adequate parallelism.

This work forms the beginning of a bridge between our research in dataflow execution and the work in parallel execution of FORTRAN. In our case, the problem is to constrain potential parallelism that is not cost effective to exploit. In the FORTRAN case, the problem is to determine where it is most cost-effective to uncover parallelism. We will never reach exactly the same place, because our analysis must err in the direction assuming two computations cannot be serialized, while their's must err in the direction of assuming two computations cannot execute in parallel. Still, we expect there will be a valuable cross-fertilization.

5.6.6 Speculative Parallelism

Richard Soley completed his Ph.D. thesis work this year on the control of speculative parallelism in Id programs, under the abstract tagged token dataflow execution model. Although resource control models for exploiting the parallelism in large scientific codes have been recently explored, no approach to exploiting speculative, searching parallelism has been explored, even though (or perhaps because) the potential parallelism of such applications is tremendous. Soley explores a view of speculation as a process which may proceed in parallel in a controlled fashion, using examples from actual symbolic processing situations.

The central issue of exploiting this parallelism is the dynamic containment of the resources necessary to execute large speculative codes. Soley shows efficient structures (graph schemata and architectural support) for executing highly speculative programs (such as expert systems) under a dataflow execution paradigm. In order to control dynamic execution graph growth, Soley develops controls over cross-procedure parallelism in an extensible manner, with applications to the various current problems of dataflow computation. Approaches to scheduling, prioritization, and search tree pruning were considered, evaluated, and compared.

In his thesis, Soley's work fleshes out the details of primitive execution resource management (function application and memory allocation), giving implementations for general and primitive resource managers and other nondeterministic constructs at the Id language level. Dynamic binding of managers is also presented to give a meaning to the term "task;" Soley's work supports the prioritization and termination of dynamically defined tasks.

The underlying constructs used by Soley's speculation control features rely on an extended definition of I-structure storage. This new definition adds an uncontrolled I-structure `WRITE` (as opposed to `STORE`) instruction, which overwrites I-structure cell contents. This nondeterministic feature is useful for implementing higher level control constructs, as Soley shows.

More revolutionary, however, is the new cell locking paradigm developed by Soley, Steele, and Barth. The new scheme is detailed in Soley's Ph.D. thesis, Steele's upcoming Master's thesis, and Barth and Nikhil's report [34]. The new locking structure of I-structure memory already implemented in the Gita simulator (by Soley and Barth) and the Monsoon prototype (by Steele), relies on the existence of structure presence bits and deferral lists to allow critical section coding of resource managers and the like. In addition to supporting busy-waiting-free lock primitives, these "dataphores" also allow the storage of data in the semaphore cell itself (hence the new name). The basic contract of the locking instructions[1] are the following:

- READ-AND-LOCK (cell): returns only when the cell has been locked, with the value written to the cell when it was allocated or last unlocked.
- WRITE-AND-UNLOCK (cell, value): unlocks the cell specified, writing the given value into the cell.

Recognizing that these instructions also support a primitive queueing mechanism (albeit of nondeterministic queue order), several other uses for this new feature have been found. MIT is pursuing a patent on this extension to dataflow (and other message passing) architectures.

5.6.7 Garbage Collection for Id on Monsoon

Arun Iyengar has begun looking at garbage collection on dataflow multiprocessors. We are implementing a copying garbage co[illegible]or for Monsoon. Simultaneously, Young is looking at compile-time techniques for detecting when heap objects are no longer needed. We plan to quantitatively study the amount of storage which can be reclaimed by garbage collection and static program analysis. We are also interested in the increased execution time and additional support required by these two different approaches for reclaiming heap storage.

5.6.8 Parallel I/O

Bhaskar Guha Roy worked on the design of a parallel I/O system for a dataflow machine. In addition to processing elements and I-structure memories, he proposed that disk units be attached to the interconnection network. Processors would interact with the disk units using split-phase transaction in a manner similar to I-structures. To initiate a disk transfer, the processor sends a token to a disk unit, specifying the direction of transfer (read/write), the address of the disk block, the address of an I-structure for the data, and the continuation of a thread that awaits the completion of the transfer. The objective is to tolerate disk latencies in exactly the same way that the latency of I-structure accesses is currently tolerated by the processor. Guha Roy designed language constructs to express parallel I/O in the presence of nonstrict data structures, and designed compilation techniques for them.

[1]Variously called lock/unlock, read-and-lock/write-and-unlock and take/put; here we shall use the most verbose forms.

5.6.9 Other Monsocn-related Work.

Ken Steele and Richard Soley proposed a design for integrating virtual memory address translation into the dataflow model [294].

Lina Muryanto and Peter Tan wrote a compiler that takes ETS code from the Id compiler and produces MC68020 code, so that Id programs may be run on Sun workstations. It uses a MIPS-like RISC language as an intermediate form, to facilitate porting it to other machines. So far, the compiler only accepts a small subset of the full language, and much work remains to be done in optimization.

5.7 Applications

We are happy to report an increase in the number of large application programs being written in Id.

5.7.1 Simulated Annealing

Stephen Brobst and Phil Kuhn implemented a number of different algorithms for simulated annealing. Simulated annealing is a heuristic that is commonly applied to a large class of opt mization problems that are known to be NP-complete, such as scheduling and building layout. They found that although the purely functional subset of Id did not lend itself well to an efficient implementation, accumulators provided an elegant paradigm for handling the nondeterministic aspects of the algorithm without sacrificing overall determinacy in the program. They also made use of Barth's lock and unlock primitives along with structure overwrites to implement a purely nondeterministic, nonfunctional version of the program. The ability to overwrite structure elements without copying the full structure provided a large reduction in the number of instructions during program execution. However, the synchronization required for correct implementation of the algorithm in the presence of structure overwrites, actually increased the critical path length of the program. Moreover, debugging and program design in the presence of locking and structure overwrite primitives became substantially more difficult. Issues of deadlock, nondeterminism, read-write races, etc. which were previously not present in the deterministic implementations became major stumbling blocks in the parallel execution environment.

5.7.2 DNA Sequence Algorithms

DNA sequence data is accumulating very rapidly. If the genetic sequence of the entire human genome is determined, databases will grow by two to three orders of magnitude from their current sizes. Parallel processing is becoming increasingly important as biological sequence data increases. Arun Iyengar implemented several different algorithms for comparing sequences using Id. Implicit parallelism makes Id a very easy language to use. One drawback is the extra copying required when an aggregate data structure needs to be updated.

5.7.3 Flight Path Generation

For his Ph.D. thesis, Michel Sadoune of the Department of Aeronautics and Astronautics has implemented a Terminal Area Trajectory Planning System for air traffic control.

A Flight Path Generator is defined as the module of an automated air traffic control system which plans aircraft trajectories in the terminal area with respect to operational constraints. The flight path plans have to be feasible and must not violate separation criteria.

The problem of terminal area trajectory planning is structured by putting the emphasis on knowledge representation and air-space organization. A well defined and expressive semantics relying on the use of flexible patterns is designed to represent aircraft motion and flight paths. These patterns are defined so as to minimize the need for replanning and to smoothly accommodate operational deviations.

Flight paths are specified by an accumulation of constraints. A parallel, asynchronous implementation of a computational model, based on the propagation of constraints, provides mechanisms to efficiently build feasible flight path plans. A network of constraints is implemented as the superposition of dataflow graphs which are synchronized distributively.

A methodology for a fast and robust conflict detection between flight path plans is introduced. It is based on a cascaded filtering of the stream of feasible flight paths and combines the benefits of a symbolic representation and of numerical computation with a high degree of parallelism.

The Flight Path Generator is designed with the goal of implementing a portable and evolving tool which could be inserted in controllers' routine with minimum disruption of present procedures.

Flight path generation and conflict detection have been implemented in Id. The program which is run with various machine configuration is composed of 600 procedures for a size of 5000 lines of Id code. It is used as a test program for the Monsoon compiler.

The conflict-free feasible flight paths which are generated and tested in an Id environment can be translated into Lisp data structures by using an interface between Id and Common Lisp. They are then displayed on the screen and simulated in an interactive manner.

5.7.4 DARPA Image Understanding Benchmark

Arthur Altman, visiting from Texas Instruments, began implementing the DARPA Image Understanding benchmark as an Id application. This benchmark performs model-based recognition of a 2 1/2 D "mobile" of rectangles from two 512 X 512 pixel images, one containing intensity data (8-bit integers), the other depth data (32-bit IEEE floating point). As such, it performs extensive numeric (data-directed) and symbolic (knowledge-directed) processing. Once the benchmark has been converted to Id, he will evaluate its potential parallelism and related performance parameters on the simulated TTDA target machine provided by the Gita environment.

5.8 P-RISC

Our work has continued to bridge the once-wide gap between von Neumann and dataflow architectures. In 1988, Nikhil and Arvind proposed a new processor architecture called P-RISC (for Parallel RISC) that properly extends a conventional RISC processor in such a way as to make it more suitable as a component for a parallel machine. The architecture will be presented at the 1989 International Symposium on Computer Architecture in Jerusalem [243].

We first organize the machine so that instruction and frame memory are local to a processor, while heap memory is global. Next, we identify a frame (activation record) as *the* register set for a thread. The control state of a thread can now be described succinctly as a token containing an instruction pointer and a frame pointer.

We now reorganize the processor so that it is multithreaded. The first step is to introduce a token queue that can contain multiple tokens. On each clock a token is dequeued and sent through the processor pipeline. The instruction it points to is fetched and executed relative to the frame that it points to. Finally, a new token is produced that is reinserted into the token queue. Note that successive tokens can be from unrelated threads.

To deal with long memory latencies, we use the technique of I-structures. A **load** instruction sends a request to memory along with a return continuation. Meanwhile, the processor is free to execute other tokens. The response from memory comes back with the return continuation—the value is stored in the frame and the continuation is requeued.

For fine-grained parallel operation, we extend the instruction set with three new instructions:

- **fork**, which is like a jump, except that it also produces the token for the next instruction (i.e., it is like a jump and continue).
- **join**, which specifies a frame offset containing a counter initialized to n, the number of threads that will execute this instruction. Each execution decrements the counter. Only the thread that decrements it to 0 continues— the other threads are discarded.
- **start**, which specifies a continuation in a different frame (which may be on a different processor), along with a value to be stored in that frame before the continuation is started.

With these instructions, it is possible to emulate the fine-grained parallelism of a dataflow graph. Being a superset of a conventional RISC instruction set, it is also possible to execute conventional compiled code, e.g., from FORTRAN.

We have begun simulation and other studies to evaluate this architecture, described below.

5.8.1 Compiling for P-RISC

Bradley Kuszmaul specified an abstract P-RISC instruction set, complete with operational semantics (specified by a relation on machine states).

He is implementing a P-RISC code generator for the Id compiler. Code generation proceeds by transforming a data flow graph into a control flow graph, performing certain optimizations, and then transforming the control flow graph into machine specific code. Examples of machine specific code which might be generated include:

- the abstract P-RISC instruction set mentioned above;
- other specific P-RISC instruction sets (such as the P-RISC co-processor for a RISC chip being worked on by Sharma, see below);
- a variant of Monsoon with registers;
- Eps'88;
- a standard serial machine (such as a RISC computer, a Vax, a Lisp machine, or a Cray supercomputer); or
- off-the-shelf parallel MIMD hardware.

It appears that the control flow graph intermediate format is well suited for the target architectures mentioned above. Currently only parts of the Id language are correctly compiled to control flow graphs, and the only machine specific code generated by the compiler are the abstract P-RISC instruction set and the serial code for the Lisp Machine. Preliminary results indicate that it may be possible to run Id programs almost as fast, i.e., within a factor of four to ten, as Lisp or C programs.

5.8.2 Simulator for P-RISC

Ira Scharf, as part of his S.B. thesis, has been working on an interpreter for the abstract P RISC instruction set developed by Kuszmaul. The objective is to build a tool like Gita, our graph interpreter for the TTDA, that has proved so invaluable in evaluating the TTDA. A first version of the interpreter is now running.

5.8.3 Implementation of P-RISC Using Ordinary RISC Processors

Preliminary to the P-RISC work, Kuszmaul and Sharma surveyed commercial RISC chips, with an eye towards P-RISC implementation. We then did some design and back-of-the-envelope analysis of various strategies for implementing P-RISC on commercial RISC hardware (possibly using some sort of co-processor to provide a hardware assist for the P-RISC specific operations). Those (very preliminary) results indicate that an unmodified commercial RISC computer might lose only a factor of 10 to 15 over a dedicated P-RISC processor.

By adding an I-structure memory to the RISC computer, the performance degradation compared to a P-RISC processor drops down to four or five (Steele has spent some effort at thinking about how to make I-structure memory work for a RISC processor). By adding hardware assist for context switching, that degradation goes down to two or three.

Sharma has been trying to identify a way of efficiently caching activation frames of tasks in the processor register set so as to minimize the penalty incurred on switching from one thread to another. We developed a write-through caching scheme that caches activation frames in a set of register windows. We have also proposed a scheme which allows a very high degree of look-ahead in the instruction stream. In other words, a processor can easily identify the next 15-20 instructions to be executed. We accomplish this by switching threads even on conditional branch instructions—which are nondeterministic in the sense that the flow of control beyond such instructions is not known until after the instruction is executed. Putting these two schemes together, we get an architecture which permits switching between threads with minimal (potentially zero) penalty. Further, the high degree of look-ahead in the instruction stream may offer several advantages that have alluded processor-pipeline designers in the past. We are currently examining these.

5.9 Functional Databases

Michael Heytens continued his investigation into the synthesis of databases and functional languages, treating an update transaction as a declarative specification of a new version of the database, inspired by the treatment of I-structures in Id. After completing the design of a kernel database language to express such updates, he has begun implementing a prototype, based on ideas from compiling Id to P-RISC machines.

5.10 Publications

[1] Arvind, D.E. Culler, K. Ekanadham. The price of asynchronous parallelism: an analysis of dataflow architectures. In *Proceedings of CONPAR 88, British Computer Society—Parallel Processing Specialists,* University of Manchester, September 1988. Also Computation Structures Group Memo 278, MIT Laboratory for Computer Science.

[2] Arvind, D.E. Culler, and G.K. Maa. Assessing the benefits of fine-grain parallelism in dataflow programs. *International Journal of Supercomputer Applications*, 2(3), November 1988. Also in *Proceedings of Supercomputing '88*, Orlando, FL, November 1988

[3] M.L. Heytens and R.S. Nikhil. GESTALT: an expressive database programming system. *ACM SIGMOD Record*, 18(1):54–67, March 1989.

[4] R.S. Nikhil. *Id (Version 88.1) Reference Manual.* Computation Structures Group Memo 284, MIT Laboratory for Computer Science, August 1989.

Theses Completed

[1] D.E. Culler. *Managing Parallelism and Resources in Dataflow Programs.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[2] S. Jagannathan. *A Programming Language Supporting First Class Parallel Environments.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, January 1989.

[3] M. Lee. *Interactions Between the Query Processor and Buffer Manager of a Relational Database System.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[4] L. Muryanto. *A Translator from ETS Dataflow Graphs into RISC Code.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[5] I. Scharf. *A Simulator for a Parallel RISC Machine.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[6] P.K.S. Tan. *A Translator from RISC Code into MC 68020 Code.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

Theses in Progress

[1] S. Aditya. *An Incremental Type Inference System for the Programming Language Id.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected December 1989.

[2] V. Kathail. *Optimal Evaluators for Lambda-calculus Based Functional Languages.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected August 1989.

Talks

[1] D.E. Culler. Assessing the benefits of fine-grain parallelism in dataflow programs. Lecture given at Supercomputing '88 Conference, Orlando, FL, November 1988.

[2] D.E. Culler. Controlling parallelism and resource usage in dataflow execution of scientific programs. Lecture given at Oregon Graduate Center, Beaverton, OR; University of Washington, Seattle, WA; University of California at Berkeley; Sun Microsystems Inc., Mountain View, CA; University of California at San Diego, La Jolla, CA; March 1989.

[3] D.E. Culler. Managing parallelism and resources in scientific dataflow programs. Lecture given at University of California at Los Angeles; University of California at Irvine, April 1989.

[4] V. Kathail. An optimal interpreter for the λ-calculus. Lecture given at Hewlett Packard Research Laboratories, Palo Alto, CA. April 1989.

[5] R.S. Nikhil. Lectures in 6.83s, MIT 1-week summer course on Dataflow Architectures and Languages, July 1988.

[6] R. S. Nikhil. Compiling graph reduction for a dataflow machine. Lecture given at the Workshop on the Implementation of Lazy Functional Languages, Aspenas, Sweden, September 1988.

[7] R.S. Nikhil. Id, a parallel programming language. Lecture given at Computer Science Day at Memorial University of Newfoundland, St. John's, Newfoundland, Canada, October 1988.

[8] R.S. Nikhil. The parallel programming language Id and its dataflow implementation. Lecture given at Boston University, November 1988.

[9] R.S. Nikhil. Implicit parallelism and heap storage are necessary for parallel languages. Lecture given at the Workshop on Opportunities and Constraints of Parallel Computing, IBM Almaden, December 1988.

[10] R.S. Nikhil. Structure and interpretation of parallel computer programs. Lecture given at Apple Computer Company, Cupertino, CA, January 1989.

[11] R.S. Nikhil. Managers: a construct for expressing state and nondeterminism in a parallel language. Lecture given at IFIP WG 2.8 (Functional Programming) Meeting, Mystic, CT, May 1989.

Information Mechanics

Research Staff

D. Kranz
N. Margolus
T. Toffoli, Group Leader

Graduate Students

M. Biafore
D. Harnanan
H. Hrgovčić
M. Smith

Support Staff

S. Hardy

Visitors

B. Chopard
F. Commoner

6.1 Introduction

In the last year we have worked hard at preparing a cogent proposal for the final design and the construction of CAM-8, a high performance cellular automata multiprocessor. The proposal has finally been funded by DARPA, and we are extremely busy now working to deliver the goods.

Some of the more theoretical research that we managed to carry out while we were waiting for an answer to our proposal, and that is described in this report, will continue at a slower pace for a while, until the design of the VLSI chip that constitutes CAM-8's "heart" is shipped to the foundry.

Our activity has concentrated on the following areas, discussed in more detail below:

- Relativistic invariance in parallel computations.
- Solid-body motion in cellular automata.
- What variational principles may look like in discrete systems.
- Further design of CAM-8—a large cellular automata machine for (mainly) physics emulation.
- Symmetric and asymmetric interface formation in conservative interactive-particle systems.
- Pattern recognition by texture-locked loop.
- Identification and experimental determination of *specific ergodicity* in invertible dynamical systems.

6.2 Relativistic Invariance in Parallel Computations

We have continued studying the problem of how relativistic invariance may emerge at the macroscopic levels in cellular automata and similar discrete systems, in which such invariance is meaningless at the microscopic level (see previous progress report).

A preliminary report on last year's work has appeared in [299]. Work is in progress to generalize those results to more than one dimension [292].

An alternate way of probing the relationship between Lorentz invariance and cellular automata is through the study of wave equation models in one or more dimensions. Note that while conventional computational models use discrete space but *continuous* state variables at each site, here we are trying to get the same behavior with *binary* state-variables.

The one dimensional version is especially tractable. Not only do we have cellular automata rules that satisfy the wave equation *exactly*; we can also evaluate in closed form, via combinatorial arguments, phase-space averages over the entire set of states of, say, a lattice string, as well as dynamical and statistical properties involving kinetic and potential energies, mean-square sums of the amplitudes of the normal modes, and the moments of the Fourier components of the system.

Because these implementations use particle-like entities to simulate wave phenomena, they also naturally lend themselves to the study of quantum mechanical phenomena, especially *vis-à-vis* Feynman path-integral methods. Generalizing our results for the n-dimensional wave equation to the corresponding Dirac and Weyl equations, we have obtained a novel lattice method for simulating single-particle quantum phenomenawave, in the spirit of the Bohm interpretation.

6.3 Solid-body Motion in Cellular Automata

This new area of work is related to that of the preceding section.

There has been much excitement over recent work on fluid modeling with cellular automata. These models have been basically *point* models: all properties of the fluids have been represented by the contents of individual cells. As one simulates a larger and larger range of material parameters, this approach requires more and more bits in each cell, with an exponential increase in the size of the lookup table (current simulations of 24-particle-per-site lattice gases, done on a CRAY-X/MP employ about *one gigabit* of fast memory for this purpose [121]!).

Viewing cellular automata as a model of fine-grained parallel computation, there are tremendous technological advantages in finding and using *simple* CA rules.

In nature, complex materials are made up of simple parts (atoms) held together in groups by various adhesive and cohesive forces. We would like to find cellular automata models with analogous characteristics: information about complex material properties should be spread out among a collection of cells. This involves the problem of simulating forces between particles in an appropriate manner (with momentum and energy conservation, and preserving reversibility) so that we can have collections of particles (bodies) moving together and interacting with one another. Since this approach provides an alternative to using extremely complex rules, we consider the problem of not knowing how to make moving bodies (and forces) in cellular automata to be a major obstacle standing in the way of extensive use of cellular automata for physical modeling.

This problem is closely related to the relativity discussion of the previous section: in a relativistically-invariant system, we have the same physics in any inertial frame. We can therefore have bulk motion of macroscopic (and microscopic) bodies, while their internal dynamics and chemistry remains essentially unchanged. Thus if we can have (collective) bodies at all in a relativistically invariant system, we automatically have moving bodies

The extent to which the opposite implication holds (moving bodies imply relativity) is an interesting question, which this research will also address.

Understanding the physical aspects of the discrete entities used in our models in order to account for deformations of solid bodies and the attendant restoring forces will lead to new intrinsic measures of physical interest, such as discrete stress tensors.

Among the many preliminary explorations we have made in this area, we have studied in some detail a simple cellular automaton model of string-like objects freely moving in space (in one, two, or three dimensions) and interacting with one another [78]. The basic object can be thought of as a chain of point-masses connected by springs; each point can have adjustable mass and momentum, and each link adjustable potential energy. Longitudinal and transverse vibrations are supported, as well as average bulk motion and collisions between objects, with strict conservation of the above quantities. Some applications of this model are under investigation [77].

6.4 What Variational Principles May Look Like in Discrete Systems

The variational principles of mechanics characterize the solutions of certain differential equations as continuous functions for *infinitesimally small variations* of which the value of certain continuous functionals remains constant.

In many "granular" dynamical systems, such as cellular automata, both the independent variables (space and time) and the dependent ones (state variables) are discrete, and thus do not admit of infinitesimally small variations. Clearly, variational principles in their traditional form cannot be employed here. On the other hand, the fundamental role played by variational principles in mathematical physics makes one suspect that something having the same flavor *should* be available in the analysis of discrete systems.

Indeed, as soon as one studies these systems from a macroscopic, combinatorial viewpoint, variational principles emerge with such regularity and strength as to make one believe that the whole approach should be reversed. Instead of taking continuous variational principles as the paragon, and looking for some imitation of them in discrete systems, one may take as a productive working hypothesis that the prototypical variational principles arise from combinatorics. Variational principles appear with such regularity in physics not because they represent some deep-seated feature of physics proper, but because they are a corollary of general *combinatorial* laws and are bound to arise whenever one considers systems consisting of a large number of elements.

We have investigated the above topics in a variety of settings, having in mind (a) the emergence of the concept of *energy* out of microscopic combinatorics, and (b) the connection between conserved quantities and symmetries. In particular, we have discovered cellular automata models displaying *exact* harmonic motion not only for infinitesimal perturbations, but also for *arbitrarily large* displacements. We have started studying the equilibrium configurations of such systems in various dissipative regimes, establishing a connection between the variational principles that govern such configurations at a macroscopic level and the combinatorics that governs them at the macroscopic level.

6.5 CAM-8—A Large Cellular Automata Machine Suited for Physics Emulation

We have completed the basic design of CAM-8, a large, high performance cellular automata machine which, for its intended areas of application, will be by far the fastest computer in the world. Indeed, this machine will constitute a "microscope" into "computational worlds" that were until now inaccessible (see previous progress report for more details), and thus will stimulate real, practical use of cellular automata as a modeling environment. Partial funding for the actual development of this machine has been granted by DARPA, starting January 1989. More conceptual aspects of this architecture, in particular in the context of simulation of stylized physical systems, fall within the scope of our NSF contract.

CAM-8 is the next generation in a line of Cellular Automata Machines (CAMs) developed at the MIT Laboratory for Computer Science, and are already used by many investigators. The essential elements of the CAM-8 architecture and some of its intended applications are reported in [232][298]. (To make the conceptual aspects and the potential applications of such machines accessible to a wide audience, we have written a book, *Cellular Automata Machines—A New Environment for Modeling*, which constitutes a comprehensive introduction to the subject and illustrates the use of an earlier machine, CAM-6, which is in commercial production. We have just completed the second edition of software and documentation for CAM-6 [71].)

The functional architecture of CAM-8 is fundamentally that of a *cellular automaton*—where a large number of identical atomic processors are uniformly interconnected to form an indefinitely extended two or three dimensional network ("polynomial interconnection" architecture) and operated in synchronism. This approach gives CAM-8 unmatched performance in dealing with discrete, fine-grained models of systems whose topology reflects that of ordinary spacetime.

Our specific implementation of the basic cellular automaton plan includes certain refinements recommended by recent theoretical developments, and makes use of a number of original solutions suggested by the current technological context. Some of these features allow CAM-8 to retain a high level of performance even in certain areas where one might expect that an "exponential interconnection" architecture (e.g., tree or hypercube) would be mandatory.

For many applications, this machine may be visualized as a volume of simulated *programmable matter* in which a large variety of experiments on spatially-extended physical systems can be performed rapidly and conveniently—a useful metaphor for this is a "silicon wind-tunnel." Other examples include the simulation of physical phenomena such as diffusion, aggregation, and phase separation; the study of properties of materials such as plasma and alloys, and of chemical reactions; the exploration of certain models of fundamental physics; and a number of practical applications such as the study of how waves propagate in a nonhomogeneous medium and are reflected by arbitrarily shaped obstacles (e.g., radar and sonar echo analysis).

In addition, CAM-8 will constitute a powerful computer for many information processing applications dealing with fine-grained structures having a high degree of regularity in at least two dimensions --for instance, a "silicon retina" with real time performance. Indeed, this appears to be an ideal architecture for many pattern recognition and tracking tasks [300].

Further, CAM-8 will provide a natural environment for the simulation of large scale logic circuits; in particular, for exploring the potential of reconfigurable circuits ("downloadable hardware").

Finally, a machine of the functionality of CAM-8 will be indispensable for designing and emulating the algorithms that will constitute the firmware of a new generation of fully parallel cellular automaton ultracomputers.

6.6 Symmetric and Asymmetric Interface Formation in Conservative Interacting-particle Systems

We have continued on interface formation in immiscible fluids, simulated on the basis of microscopic first principles. In brief, we study the approach to equilibrium of a system consisting of a large number of particles of two kinds, coupled by local interactions, under the constraints of strict invertibility and conservation of particle species and total energy.

To this end, we have equipped one CAM-6 unit with extended processing tables, capable of handling the more complex local interactions required by these models. Besides refining and extending work done in this field by others on symmetric interface formation—which entails only *binary* interaction—we have started exploring the more challenging area of *multiplet* interactions, which has allowed us to model, among other things, *asymmetric* surface tension effects.

The necessary energy bias in the rule due to curvature was recognized to be an approximation to a new distributed quantity that we have called *winding number density*, and the ramifications of this quantity are being investigated. These studies have also yielded benefits by suggesting the development of graph related algorithms and topological concepts for cellular automata [291].

6.7 Pattern Recognition and Tracking by Texture Locked Loops

We have continued working on a pattern recognition method that is applicable to a limited but pervasive class of patterns typically, natural landscape features such as rivers, coastlines, urban agglomerations, etc., as they may appear on a satellite photograph; fingerprints and other biological constructs; and, in general, textures and structures whose long range spatial correlations are ultimately explainable in terms of the repeated action of simple local mechanisms [300]. This method, which can be thought of as a generalization of the well-known *phase-locked loop*, is insensitive to large amounts of noise; it can take good advantage of the peculiar tradeoffs in computational resources offered by fine-grained parallel processors (such as Cellular Automata Machines, the Connection Machine, and the Massively Parallel Processor); finally, by its very nature, this method is to a certain extent "aware" of its capabilities and its limitations.

This approach to pattern recognition has been presented to several technical audiences (GE Research Labs, Naval Laboratory, and Lincoln Labs) and has been well received. We are now exploring specific applications.

6.8 Specific Ergodicity

We have started working on a new theme, mainly the identification of a new quantity of interest in dynamical systems, and its experimental determination in a number of representative cases.

Specific ergodicity asks, for an invertible cellular automaton, what fraction of the total information needed to identify an individual state is devoted to specifying the position of this state on its orbit. We give empirical evidence that this question has a definite answer. A preliminary report on this work appeared in [299].

The experiments reported in the above reference were performed using a few AT clones full time for several weeks, which is equivalent to several hours on a typical supercomputer. In view of the exponential complexity of the problem, even moderate improvements on the numerical estimates obtained so far would require a drastic increase in computing power. For simple cellular automata, a speedup of $\approx 10,000$ can be achieved with a dedicated, fully parallel implementation consisting of a few programmable gate-array chips. We have set up fast dedicated simulators of this kind for some simple two dimensional cellular automata, using the largest available XILINX chips, and we are beginning to collect experimental data.

6.9 Publications

[1] C. Bennett. Notes on the history of reversible computation. *IBM Journal on Research and Development*,32:16–23, 1988.

[2] C. Bennett, N. Margolus, and T. Toffoli. Bond-energy variables for Ising spin-glass dynamics. *Phys. Rev. B*, 37:2254, 1988.

[3] A. Califano, N. Margolus, and T. Toffoli. *CAM-6 User's Guide* (second edition), Systems Concepts, San Francisco, CA, 1989.

[4] B. Chopard. Strings: a cellular-automaton model for moving objects. *Proceeding of a Workshop on Cellular Automata and Modeling of Complex Physical Systems*, Les Houches, Springer-Verlag, 1989.

[5] B. Chopard. In preparation.

[6] N. Margolus. *Physics and Computation.* Technical Report MIT/LCS/TR-415, MIT Laboratory for Computer Science, 1988.

[7] N. Margolus and T. Toffoli. Cellular automata machines. Revised version of a paper of the same title appeared in *Complex Systems*, 1:967–993, 1987. Also to appear in *Large Nonlinear Systems*, G. Doolen, editor, 1989.

[8] N. Margolus. Cellular automata machines—a new environment for modeling. In *Proceedings of the 1988 Rochester Forth Conference*, L. Forsley, editor, Institute for Applied Forth Research, 1988.

[9] Staff writers. Logic for conservatives. *The Economist*, 78–79, February 11, 1989.

[10] U. Frisch, et al. Lattice gas hydrodynamics in two and three dimensions. *Complex Systems*, 1(4):633–707, 1987.

[11] H. Hrgovčić. A new lattice model for quantum-mechanical behavior. In preparation.

[12] M.A. Smith, H. Hrgovčić, N. Margolus, and T. Toffoli. A conformally invariant cellular automaton. In preparation.

[13] M.A. Smith, N. Margolus, and T. Toffoli. Cellular automaton simulations of oil-and-water mixtures. In preparation.

[14] M.A. Smith. A fermi lattice gas in a potential well. In preparation.

[15] M.A. Smith. Cellular automata in physics education. In preparation.

[16] T. Toffoli. Information transport obeying the continuity equation. *IBM Journal on Research and Development*, 32(1):29–36, January 1988.

[17] T. Toffoli. Pattern recognition and tracking by texture-locked loops. Working paper, to appear in expanded form as a chapter of *Advanced Computer Architectures for Robotics and Machine Intelligence: Neural Networks and Neurocomputers*, IEEE, 1988.

[18] T. Toffoli. Cellular automata machines as physics emulators. In M. Budinich, et al., editors. *Impact of Digital Microelectronics and Microprocessors on Particle Physics*, pages 154–160, World Scientific, 1988.

[19] T. Toffoli. Four topics in lattice gases: ergodicity; relativity; information flow; and rule compression for parallel lattice-gas machines. In *Proceedings of Discrete Kinetic Theory, Lattice Gas Dynamics and Foundations of Hydrodynamics*, Institute for Scientific Interchange, pages 20–24, Turin, Italy, September 1988.

Theses in Progress

There are three PhD theses in progress.

Mercury

Academic Staff

B. Liskov, Group Leader	W. E. Weihl

Research Staff

T. Bloom	P. Johnson
D. Clark	K. Sollins
D. Curtis	J. Wroclawski

Graduate Students

E. Waldin	S. Zanarotti

Undergraduate Students

S. Prager

Support Staff

A. Aldrich	G. L. Staton
A. Rubin	

Visitor

L. Shrira

7.1 Introduction

Mercury is a communications mechanism that supports efficient communication among program modules in a distributed, heterogeneous environment [212][213]. Modules act as clients and servers: a server is a module that provides a number of procedures that can be used by other modules, called *clients*, to interact with it. Communication occurs by means of *call streams*; clients make calls to the procedures provided by servers over these streams. A client is able to make three kinds of calls: synchronous calls, in which the client waits until the call returns before making a subsequent call on that stream; asynchronous calls, in which the client can make a number of calls on the stream without waiting and pick up the results of the calls later; and sends, which are like asynchronous calls except that the client picks up results only if a call terminates in an exceptional condition.

During the current year, we have continued to work on the design and implementation of the Mercury communication mechanism. In addition, we have developed a new protocol for implementing at-most-once messages efficiently, and have started work on a new project to provide an object repository for use in a heterogeneous network.

7.2 A Formal Specification for Mercury Call Streams

B. Liskov and L. Shrira have provided a formal specification for Mercury call streams. An important goal in specifying Mercury streams is to allow the different language veneers to present streams differently to their users. The intention of the specification is then twofold: first, to guarantee that the different veneers "understand" each other; and second, to limit the information exposed by the veneer operations. The specification deals with the safety properties of the protocol; it does not address the performance aspects of call streams.

The specification uses an event-based model. It defines the *events* that can be observed by users of streams and restricts the legal sequences of those events. The specification allows differences in veneers by defining the common primitive events that underlie the veneer operations. In other words, the specification does not define a set of stream operations that all veneers must provide. Instead, veneers are free to define a convenient set of operations. However, the meaning of any stream operation provided by a veneer must be defined in terms of a legal sequence of the defined events that represents the effect of that operation on the stream. For example, in one veneer, user programs at servers might explicitly wait for the next call to arrive; while in another, a process might be created automatically by the veneer when a call arrives without the user code having to wait. The executions of the stream operations in both veneers must be explained using event sequences permitted by our specification.

The operations in different veneers need not expose all details of streams. The specification defines the most that can be observed by user code; veneers are always free to hide detail. For example, user code at the receiver may not be able to observe that a stream is broken (i.e., unable to transmit messages) even though our events convey this information.

7.3 Mercury/Argus Veneer

T. Bloom and D. Curtis have been working on the implementation of the Argus veneer. Stream calls have been added to the client-side veneer. The promises mechanism [213] is implemented with a procedural interface rather than syntactic support. The Mercury catalog implementation has been enhanced to provide both service and port registration, and the restart/recovery mechanism is fully implemented. At this point the Argus veneer is fully functional, with the exception of support for a few additional Mercury types (vspaces) to be incorporated. Work has started on designing a test suite for use with all the veneers and on performance analysis. D. Curtis and her students have implemented a calendar application and a distributed login-uid as Mercury services are built in Argus.

7.4 Transactions in Mercury

B. Liskov and W. Weihl have developed a design for the protocols to be used in messages concerning transactions in Mercury. Mercury transactions will be compatible with those in Argus so that Argus servers can be accessed under Mercury. However, some changes from the Argus protocols are needed because the constraints in Mercury are somewhat different than Argus. For example, in Argus every remote call must be made as a subaction; Mercury does not require this. Making a call as a subaction insulates the caller from failures that occur in the call: if the call does not complete, or it aborts at the callee, the calling action need not abort. If there is no subaction, these circumstances will force the calling action to abort, and the messages exchanged in this case must indicate this fact.

B. Liskov and W. Weihl also defined a new entity called a *transaction management server*. Such servers will run at many Mercury nodes and can be used across the net via Mercury streams. A transaction management server performs various housekeeping chores associated with transactions on behalf of clients. It is advantageous for two reasons:

1. It reduces the work needed to implement transactions in C and Lisp. Instead these veneers can call on the server to do much of the work in implementing transactions.
2. It can reduce the probability of a failure occurring in the middle of two-phase commit. This is possible because the servers can be located at more reliable nodes, and will not be directly under the control of users who might, for example, turn off the machine they are using.

The servers themselves are easy to implement, since they are simply specialized Argus guardians. Also, the server design does not require extra communication that would delay the execution of user transactions. For example, it is not necessary to communicate with the server to create a transaction or to make a call. Instead, the server is used only at two-phase commit and delays the commit of the transaction as seen by the user by one message delay.

7.5 Efficient At-most-once Messages Based on Synchronized Clocks

B. Liskov, L. Shrira and J. Wroclawski [214] have designed a new efficient message passing protocol that guarantees at-most-once message delivery without requiring communication to establish connections. The goal is to be able to accept messages most of the time even when the receiving module has no state information stored about the sending module. The scheme is interesting because it allows us to efficiently implement at-most-once remote procedure calls (RPCs), even when there are large numbers of clients and servers and when clients communicate with servers only occasionally.

At-most-once semantics for RPCs means that a call is guaranteed to be executed at most once even when failures occur such as a crash of the receiving module. It is desirable because it provides proper semantics even when calls are not idempotent. However, the implementation of at-most-once semantics can be expensive because the server needs a way of determining whether it has seen a message before. The determination can be made if the server maintains some state, known as a *connection*, for the client. If there is no state, the connection must be established, which typically requires a pair of messages to be exchanged between the client and the server. If the connection is used for many calls, the cost of the connection setup can be amortized across all of them. If there are only a few calls, the overhead is high relative to useful work. In the worst case, only one call will be made on the connection, and the cost of the call is doubled. Yet this case may be quite common; it corresponds to clients using servers only occasionally.

To avoid the cost in this common case, systems have provided *at-least-once* semantics, which provides only weak guarantees about how many times a call is executed. For example, even when a call terminates normally, it may have been executed more than once. Some systems provide at-least-once semantics as the only option; others provide it as an alternative available to the client if desired. Both approaches are undesirable: with only at-least-once available, the application programmer must cope explicitly with the problems arising from non-idempotent calls. Things are better when both are available, but the communication system is more complicated than if there is just one choice.

Our work shows that it is practical and efficient to provide only at-most-once semantics. The method allows calls to be made without prior communication to establish a connection. Ours is not the first method to do this; the Delta-t protocol [307] also avoids connection setup. However, we use a different technique based on loosely synchronized, monotonic clocks. Our protocol can easily tolerate the clock skews provided by existing clock synchronization protocols; these skews are typically less than 100 milliseconds. If the rare event of unsynchronized clocks does occur, the protocol continues to work correctly although there is a degradation of performance. The protocol requires that clocks at servers that survive crashes be monotonic; it does not rely on properties of clients' clocks for correctness.

We used the message protocol to implement at-most-once RPCs based on the SunRPC library and compared our performance with at-least-once and at-most-once RPCs already available in the SunRPC library. Our performance measurements indicate that at-most-once RPCs can be provided at the same cost as less desirable ones that do not guarantee at-most-once execution.

7.6 Object Repository

B. Liskov and L. Shrira and a group of students have been working on the design of an Object Repository. Programs of today make use of file systems to store data that must survive from one day to the next. Programs of the future will use object repositories instead. Object repositories are better suited to the needs of programs and users because they correct several deficiencies of file systems, as discussed below.

Our repository will provide the following features: it will be language independent, store typed objects, support atomic transactions, be both highly reliable and highly available, and will control access to its objects. The repository fits in well with the work on Mercury, which provides a method for clients to use repository through its call streams and provides a language-independent type system that can be used in the repository. It will also be a useful service to be made available through Mercury.

Some aspects of the repository are:

Objects vs. Files: An object repository stores objects instead of files. Objects differ from files in three important ways. First, they are often small. File systems tend to be biased toward large objects, so that users must combine small objects together into large ones to use the system efficiently. By contrast, an object repository must be engineered to work efficiently for small objects as well.

Secondly, an object repository knows about the types of objects stored in it. File systems do not have such information, so they provide no help for users to avoid type errors. An object repository does provide such help. Of course, the types in use in the repository must be independent of particular programming languages, since we want to allow many different languages to use, and communicate through, the repository. We already have such a type system, namely, the type system developed for communication in Mercury. Furthermore, the Mercury library can be relied upon to store information about types, and to give a system-wide meaning to types, thus permitting us to avoid type errors in the object repository.

The type system for the object repository must includc abstract data types because the repository will need to invoke operations of a type when processing queries. The operations must be defined by the person who defines the new type.

The third point is that objects in object repository may refer to other objects in the repository, thus representing the kinds of sharing structures that are often useful in programs. By contrast, file systems do not allow files to refer to other files in a way understood by the system. Having interconnected objects raises interesting questions related to naming the objects, reading complex objects, allocating and deallocating memory for objects, and garbage collection.

Transactions: Interactions with the object repository will occur within atomic transactions; we expect to use the Mercury mechanism here. However, an object repository is likely to use a transaction mechanism in a nonstandard way. For example, a program development support system might be implemented on top of an object repository. When a person is working on a new release of a particular module, he is likely to have that object locked for a long time. We would not want to provide such an ability by running the entire production of the new release as a transaction, since holding locks for a long time is not good for system performance. Instead we need to devise a different kind of interface, probably of the "check-out/check-in" variety. The design of such an interface is a challenging problem.

Availability and Reliability: The repository must be both highly available, so that individual objects are very likely to be usable when needed, and highly reliable, so that information entrusted to it is not lost with high probability. The plan is to store objects in the repository at a small number of server nodes. To speed up interaction with the repository, clients will maintain caches containing recently used objects. To achieve high availability, we will need to use replication. Our new primary copy technique [246] should be good for the repository. Copies of each object will reside at several servers; the database as a whole will be partitioned so that the load at the servers that implement the repository will be balanced.

If the servers that store copies of an object are sufficiently failure independent, then a highly available system is also highly reliable. We may choose to achieve failure independence by equipping our servers with universal power supplies. In addition, we plan to investigate both archival mechanisms and backup mechanisms to be used when catastrophies occur.

Access Control: Sharing is only useful if it can be controlled. For example, sensitive data will be stored in the repository only if reading can be controlled. Such control can be achieved by access control mechanisms based on the authentication methods being developed for Mercury.

Language Interface: T. Bloom and S. Zdonik (of Brown University) have been working on issues of object-oriented database design. They have been looking at the problem of merging object-oriented languages and databases into seamless database programming languages with uniform access to all objects. This work is related to the way the object repository will interface to Mercury host languages.

7.7 Publications

[1] B. Liskov, L. Shrira, and J. Wroclawski. Efficient at-most-once messages based on synchronized clocks. Submitted for publication.

Theses Completed

[1] S. Prager. *A Library for Mercury, an Inter-Language Communication System.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

[2] S. Zanarotti. *Naming in an Archive Service.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

Talks

[1] J. Wroclawski. Distributed system communication and high speed networks. Lecture given at Network Computing Forum, Ann Arbor, MI, June 1989.

Parallel Processing

Academic Staff

R. Halstead, Group Leader

Research Staff

D. Kranz

Graduate Students

L. Bagnall	M. Ma
J. Loaiza	D. Nussbaum
R. Osborne	

Support Staff

S. Hardy	F. Ugwuegbu

Visitors

E. Mohr	I. Vuong-Adlerberg
H. Takagi	

8.1 Introduction

The 1988-89 academic year marked the final year of operation of the Parallel Processing Group, due to the group leader's departure from MIT.

The group's focus has been to learn how to build parallel processors that can be programmed for general purpose applications. The group's efforts are based on the parallel Lisp language *Multilisp* [147][149][148]. Members of the group have worked on several aspects of parallel processing: implementation of parallel Lisp systems, speculative computation, applications for parallel Lisp, parallel program debugging and tuning aids, and design of architectures well suited for parallel Lisp.

A major milestone during the year has been the completion of the Mul-T high performance parallel Lisp system [189], which compiles code for the Encore Multimax multiprocessor and largely obsoletes the group's earlier (interpreter-based) Multilisp implementation hosted on the *Concert* multiprocessor [147][149][152]. Being smaller and more malleable, the Concert implementation of Multilisp continues to be useful for quick experiments with modifications of Multilisp, but the Mul-T system has performance that is better by two orders of magnitude.

Other milestones include the successful demonstration of a Multilisp system that supports speculative computation and the completion of *ParVis*, a tool for debugging and tuning parallel Lisp programs. Investigations of the parameters affecting performance of parallel Time Warp [172] simulations, and of naming problems in Lisp systems, were conducted. Finally, a preliminary version of *MARCH*, a processor architecture capable of efficiently executing parallel Lisp programs, was studied via simulation, and directions for future improvement were identified. A notable result of this study was the design of control logic for a coherent cache that can connect to a multithreaded processor and a split-transaction bus.

The following sections describe each of the above mentioned aspects of the group's activity in more detail.

8.2 High Performance Parallel Lisp

A major accomplishment for the year was the completion (by D. Kranz and E. Mohr) of the Mul-T high performance parallel Lisp system [189], which runs on an Encore Multimax multiprocessor. This effort began with the T system from Yale, which implements a dialect of the Scheme language [1][272]. Mul-T is a parallel version of T with a parallel garbage collector. Mul-T is most notable for its high performance: use of the T system's ORBIT compiler [190][188] leads to performance about 100 times faster than the group's earlier Multilisp implementation on the Concert multiprocessor.

Mul-T shows that Multilisp's **future** construct can be implemented cheaply enough to be useful in a production-quality system. On the Boyer Lisp benchmark, for example, Mul-T was able to achieve higher performance even on two processors than is achieved by running the sequential Boyer program in the sequential T system (whose compiler is as good as or better than competitive commercial Lisp compilers) [189]. In addition to its performance advantages, Mul-T includes a set of debugging functions that extend T's debugging features to handle a parallel execution environment.

Innovations in Mul-T include the use of *inlining* as a run-time mechanism to increase task granularity and reduce task-management costs, and the concept of *groups* of tasks which provide convenient units to manipulate when debugging. Mul-T has been made available at no charge through network file transfer, or at a nominal charge from Encore Computer Corporation.

8.3 Speculative Computation

Our investigation of the design of mechanisms to support *speculative computation* in Multilisp continues. Speculative computation is eager evaluation where the result(s) of the evaluation may be unnecessary. It is a gamble whereby one trades additional, possibly unnecessary, computation for potentially faster execution. Speculative computation contrasts with *mandatory computation*, in which all computations are presumed to be necessary. Speculative computation requires a means to control computation to favor the most promising computations, and the ability to abort computation and reclaim computation resources.

Our interpreter-based implementation of Multilisp [147][149][148] has been extended (by R. Osborne) with constructs for speculative computation, and performance of several speculative programs has been measured. These measurements demonstrate that performing computations in parallel before their results are known to be required can yield performance improvements over conventional approaches to parallel computing. On the Boyer theorem-proving benchmark and a traveling-salesman application, speculative computation yielded performance improvements of up to a factor of 2 over the best program using only mandatory constructs, while at the same time eliminating the tuning of parameters needed to achieve that performance in the mandatory arena. On a heuristic program to solve the 8-puzzle [249], a performance increase of a factor of 26 was measured.

The main conceptual contribution of this work is a *sponsor model* that provides a framework for management of speculative computation. This sponsor model handles control and reclamation of computation in a single, elegant framework. A *sponsor* is an agent that controls the allocation of resources to computation. A sponsor supplies *attributes* (such as a priority, or a claim on a certain quantity of compute time) to computations that are sponsored by it. Every running task is sponsored by one or more sponsors, and every sponsor can sponsor several tasks, as well as sponsor other sponsors. Thus, sponsors can be organized into hierarchical networks that mirror the structure of the computation, and sponsors are a modularity construct that provide a "handle" on a subcomputation that can be used without explicit reference to the set of tasks currently performing that subcomputation. The contribution of the current work is the development of the basic sponsor model (inspired by the work of W. Kornfeld and C. Hewitt [186]) into a concrete mechanism, illustration of how it can be implemented with acceptable efficiency, and demonstration of its application to speculative computation scenarios including side effects and complex inter-task dependencies.

8.4 Naming in Lisp Systems

A separate exploration (by J. Loaiza) focused on the problem of defining and creating modules in Lisp. Three general styles of specifying and creating modules were identified. The three styles vary in ease of use, simplicity, and modularity. A module-system design was developed that treats a module as an independent object that requires a set of input values and produces a set of output values. The code that interconnects modules is kept separate from the code that implements modules. Methods of making dynamic modifications to a system of modules were also explored.

8.5 Analysis of Scheduling in Parallel Simulations

Analysis of a message-based system for concurrent simulation in Multilisp, based on the Time Warp system of D. Jefferson [172], was completed (by M. Ma). Time Warp is based on the paradigm of *tasks* exchanging *messages*. Each message has a "virtual," or simulated time; messages received by a task are to be processed in order of increasing virtual time. Time Warp uses an optimistic concurrency control strategy in which processors eagerly process available messages even when it is possible that a given message will be processed before another message with a lower virtual time but a later real time of arrival at its destination task. When this occurs, the destination task must be *backed up* to an earlier state and re-executed from that point so that the required virtual time order is not violated. Backups clearly represent wasted work and should be minimized. The number of backups is thus an important performance measure, along with the total time taken to execute a simulation.

Parameters that affect the performance of our simulation system were investigated using various simulations, including digital circuits, a queuing system, and simulations of message flow in communication networks with grid and butterfly topologies. The parameters investigated include different choices of scheduling algorithm, based not only on whether the scheduling algorithm was static, dynamic or nondeterministic, but also on the method of grouping tasks into "partitions" (scheduling units). Certain methods of partitioning yielded much better results (processing time and number of backups) than others. A model describing the underlying phenomena that govern the performance of Time Warp simulations was developed and evaluated. A key predictor of performance is the extent to which the scheduling method chosen succeeds in minimizing the variations in virtual times between partitions (variations within a partition were not so important).

8.6 Parallel Program Development Aids

A difficult and important problem in programming parallel processors lies in understanding the behavior of programs. Examples include determining where time is being spent in various parts of a program, which parts are being executed in parallel, and where the bottlenecks are located. L. Bagnall completed development of a tool, ParVis (*Par*allel *Vis*ualization), for visualizing the execution of Multilisp programs.

During a Multilisp program run, ParVis records the time of events representing task state transitions and intercommunication, such as task creation, blocking, resumption, and termination. ParVis can then generate a graphical display of this information. The system provides a display with interactive features such as scrolling and zooming, which allow the user to examine various parts of the display.

Because the program visualization utility is not interactive, but creates a display after a program run, ParVis communicates via data files rather than via an interactive network connection. Implementations of both Multilisp and Mul-T have been modified to generate the necessary trace data files.

Because the additional information described by ParVis can lead to large, complex displays, a filter language is provided to allow the programmer to specify the displayed items that are of interest. Particularly notable is the interface between the filter language and the graphical display, which allows easy inclusion of display elements in filter definitions.

ParVis has already been used extensively by several group members to find performance bottlenecks and analyze the effect of different scheduling disciplines (e.g., for speculative computation) on performance.

8.7 Architecture for Parallel Processing

An effort to learn how innovations in parallel architectures could improve the performance of parallel Lisp programs is being pursued (by R. Halstead, D. Nussbaum, H. Takagi, and I. Vuong-Adlerberg) through the development and evaluation of a processor architecture called *MARCH* (*M*ultilisp *ARCH*itecture). This project began in the previous year with the specification of *MASA* [150], a processor architecture inspired by the HEP-1 [187] and SPUR architectures [162][297]. MARCH differs from MASA in a number of details concerning procedure linkage, trap handling and task management, which have been revised in light of experience with writing run-time support routines for MASA.

Like MASA, MARCH features several non-overlapping register sets that can be used in a manner like SPUR's register windows to reduce memory accesses associated with procedure invocation. Alternatively, register sets can be allocated to concurrently executing tasks assigned to the same processor. MARCH thus aims to make procedure linkage efficient and make task creation equally efficient. MARCH's pipeline, like that of the HEP-1, can be filled by issuing instructions from different tasks on consecutive clock cycles. The fast context switching implicit in the HEP-style instruction issue should also help bridge memory access latencies by allowing other tasks' instructions to be executed while awaiting completion of a memory operation.

Like SPUR, MARCH depends on software trap handlers to handle infrequent conditions and manage processor resources such as register sets. In particular, a *scheduler* must manage the mapping from the (unbounded) set of runnable tasks to the (finite) set of register frames, and a *frame saver* must be invoked whenever a request for a register frame is made and there are currently no free frames to satisfy it.

During the year, implementation of a MARCH simulator was completed, the ORBIT compiler from the Mul-T system was retargeted to generate code for MARCH, and basic versions of the frame saver, trap handlers, and scheduler were written. As a result, it became possible to run actual (but small) parallel Lisp application programs and measure the performance effects of different design decisions. The need for improved scheduling performance led to the definition (by H. Takagi) of an interprocessor interrupt mechanism for MARCH. Experiments (by H. Takagi) showed that grouping the scheduling and frame saving functions into a separate *housekeeper* task generally yields better performance than invoking those functions via traps.

As a result of the year's activities, considerable progress has been made in discovering how to achieve MARCH's initial goal of efficient execution of parallel Lisp programs, but work is still needed to reduce several performance costs. Notably, MARCH's HEP-like instruction-issue mechanism only issues an instruction when the previous instruction in the same instruction stream has completed—this requires a large number of streams to fully utilize a pipelined processor. Also, housekeeping costs are still too high. Although the Parallel Processing Group will not continue in existence at MIT, we hope that some of the unfinished work on MARCH may continue as part of the APRIL project [9].

Another architectural project (pursued by I. Vuong-Adlerberg) concerned the design of a coherent cache for MARCH. The cache is based on a shared bus, which is made a split-transaction bus for increased bandwidth. MARCH's ability to interleave several independent instruction streams prevents the processor from systematically becoming blocked on every cache miss, but presents new challenges for cache design.

Upon a cache miss, a conventional cache remains unavailable for further processor requests until the data has been fetched and the miss has been completely processed. A new cache design is needed for a multithreaded processor like MARCH, which will allow the processor to continue to access the cache even while misses are being processed. The use of a split-transaction bus further complicates the design challenge by increasing the variety of asynchronous events to which the cache may need to attend. A cache design to meet this challenge was developed [305], the kind of coherence that it provides was formally defined, and the validity of the cache design was proven. The literature offered very little pre-existing support for such proofs, so a formalism was also developed for expressing statements about consistency and aiding in their proof [305].

8.8 Publications

[1] E. Bradley and R. Halstead. Simulating logic circuits: a multiprocessor application. *International Journal of Parallel Programming*, 16(4):305–338, August 1987.

[2] R. Halstead. Design requirements for concurrent Lisp machines. In K. Hwang and D. DeGroot, editors, *Parallel Processing for Supercomputers and Artificial Intelligence*, pages 69–105, McGraw-Hill, 1989.

[3] D. Kranz, R. Halstead, and E. Mohr. Mul-T, a high performance parallel Lisp. In *ACM SIGPLAN '89 Conference on Programming Language Design and Implementation*, pages 81–90, Portland, OR, June 1989,

[4] T. Sterling, A. Musciano, D. Becker, and R. Osborne. Multiprocessor performance measurement using embedded instrumentation. In *International Conference on Parallel Processing*, August 1988.

Theses Completed

[1] L. Bagnall. *ParVis: A Program Visualization Tool for Multilisp.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, February 1989.

[2] J. Loaiza. *Naming as a Means of Organizing Large Systems, or a Modular Naming System for Lisp.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, August 1988.

[3] M. Ma. *Efficient Message-based System for Concurrent Simulation.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, January 1989.

Thesis in Progress

[1] R. Osborne. *Efficient Support for Speculative Computation in Multilisp.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected October 1989.

Talks

[1] L. Bagnall. ParVis: a program visualization tool for Multilisp. Lecture given at Workshop on Parallel Computer Systems: Performance Instrumentation and Visualization, Santa Fe, NM, May 1989.

[2] R. Halstead. Parallel symbolic computing using Multilisp. Lecture given at IBM Corporation, Hawthorne, NY, August 1988.

[3] R. Halstead. Parallel symbolic computing using Multilisp. Lecture given at DEC Systems Research Center, Palo Alto, CA (September 1988); Apple Computer Corporation, Cupertino, CA (September 1988), New York University, New York (March 1989), MIT Laboratory for Computer Science (May 1989), NEC Corporate Research Laboratory, Tokyo, Japan (June 1989).

[4] R. Halstead. The Multilisp project: lessons and current directions. Lecture given at U.S.-Japan Workshop on Parallel Lisp, Tohoku University, Sendai, Japan, June 1989.

[5] R. Halstead. Parallel symbolic computing using Multilisp. Lecture given at Tohoku University, Sendai, Japan (June); Electrotechnical Laboratory, Tsukuba, Japan (June); NTT Corporate Research Laboratory, Tokyo, Japan (June), 1989.

[6] R Halstead and D. Kranz. Implementing an efficient parallel lisp. Lecture given at Workshop on Compilation Techniques for Parallel Computing, Cornell University, Ithaca, NY, August 1988.

[7] Osborne, R. Speculative computation in Multilisp. Lecture given at DEC Cambridge Research Lab, Cambridge, MA (February); Carleton University, Ottawa, Ontario, Canada (February); University of Waterloo, Waterloo, Ontario, Canada (March); University of Victoria, Victoria, British Columbia, Canada (March); Simon Fraser University, Burnaby, British Columbia, Canada (March); U.S.-Japan Workshop on Parallel Lisp, Tohoku University, Sendai; Japan (June), McGill University, Montreal, P.Q., Canada (June), 1989.

Programming Methodology

Academic Staff

B. Liskov, Group Leader	W. E. Weihl

Research Staff

D. Curtis	P. Johnson

Graduate Students

B. Ben-Zvi	Q. Huang	G. Leavens
J. Cohen	D. Hwang	S. Markowitz
S. Ghemawat	A. Joseph	S. Perl
R. Gruber	R. Ladin	C. Waldspurger
W. Hsieh		

Undergraduate Students

A. Farrell	C. Maeda
J. Chun	B. Spiers
G. Curwin	R. Stata
M. Levine	

Support Staff

A. L. Rubin	A. Wiseman
L. Sprung	

Visitor

L. Shrira

9.1 Introduction

Research in the Programming Methodology Group has continued to focus on the area of distributed computing. In addition to our work on Argus and Mercury, we have also studied replication methods, implementation of distributed applications, and theory of distributed systems. Our research in these areas is described below.

9.2 Argus and Mercury

We have continued our study of how to extend Argus to provide access to Mercury mechanisms. One issue is how to relate Argus types to Mercury types. The mechanism must support two activities: building relationships between Argus and Mercury types, and indicating what relationships to use in making a remote call. Ideally, the method chosen must make it easy to do things in a standard way, yet make it possible to relate types in nonstandard way.

Building a relationship is done by defining an *association*, which contains two translation functions, one mapping from an Argus type to a Mercury type and the other mapping in the opposite direction. Typically, an association will be defined as part of implementing an abstract type, although it is possible to define one independently as well. Argus will provide a number of builtin associations that relate builtin types to Mercury types. No restrictions are placed on the number of associations for a type; instead, an Argus type can be associated with many Mercury types and vice versa. For each Argus type, one association can be declared the default.

Indicating what associations to use in making a call is done as part of the declaration of the type of the remote procedure being called. If a default association is desired for a particular parameter, only the Argus type need be given for that parameter. Otherwise, the association to be used is indicated explicitly. For example,

h: **handlertype** (char) **returns** (int$to_int16) **signals** (over(real))

indicates that in calls of h, the default association should be used for the character argument, and also for the real result in the case where the exception *over* is signaled. However, if the call returns normally, the association *int$to_int16* should be used to map the 16 bit integer returned by the call into an Argus int. (The default association for ints maps them to 32 bit integers.)

9.3 Formal Models for Nested Transactions

William Weihl, working with Nancy Lynch, Michael Merritt, and Alan Fekete, has continued working on formal models for nested transactions. In the last year, we have completed a draft of a book, the goal of which is to unify and generalize the work done over the past several years on modeling algorithms for transaction processing in distributed systems. Our attempts at unification have resulted in proofs of a number of interesting algorithms, all in the same general framework. The development of the general model has also resulted in the invention of new algorithms that generalize and extend existing algorithms, both to handle nested transactions and to permit more concurrency than is permitted by existing algorithms. We have also defined correctness conditions for implementations of atomic data types in languages such as Argus. These correctness conditions serve as useful guidelines during program design and implementation, in much the same manner as loop invariants can be used for sequential programs.

9.4 Recovery Algorithms

William Weihl has continued work on formal models and verification techniques for recovery algorithms for transaction systems. Most previous work treats concurrency control and recovery as independent problems. In practice, however, designing a concurrency control algorithm requires careful consideration of the details of recovery. We have developed a model for transaction processing systems that allows concurrency control and recovery algorithms to be described abstractly in simple mathematical terms. The interactions between concurrency control and recovery can then be analyzed relatively simply. In a separate step, the implementations of the concurrency control and recovery algorithms can each be shown to implement the more abstract descriptions used to analyze their interactions. We have used the model to analyze the constraints placed by two separate recovery algorithms, update-in-place and deferred-update, on conflict-based concurrency control algorithms. We have proved necessary and sufficient conditions for a concurrency control algorithm to work with each of the recovery algorithms. These conditions are interesting for several reasons. First, they give precise bounds on the level of concurrency permitted by each recovery method. Second, they directly lead to new concurrency control algorithms that permit more concurrency than previously existing algorithms. Third, the two recovery algorithms are incomparable in terms of the constraints each places on concurrency control: each permits concurrency control algorithms that the other does not. These results are described in a paper in the *Proceedings of the 1989 Symposium on Principles of Database Systems.*

9.5 Storage Management for Persistent Memory

William Weihl and Elliot Kolodner have been working on efficient automatic storage management for persistent memory. Crash recovery algorithms for databases require explicit interaction with the recovery system to allocate and free stable objects, and do not cope with objects that change locations. We have developed garbage collection algorithms for crash-tolerant systems. The algorithms are described in a paper in the *Proceedings of the 1989 SIGMOD conference.* We are currently looking at incremental and generation-based methods.

9.6 The Meaning of Subtypes

In a thesis finished in December 1988 [199], Gary Leavens describes a verification method for object-oriented programs that use subtypes. Object-oriented programs are polymorphic, in the sense that the type (or class) of the target of a message may not be known until run time; thus, the actual code that will be run as the result of an invocation is not statically determinable. Leavens presents the idea of a "simulation relation," which explains how the objects of a subtype can be viewed as objects of a supertype, and shows how simulation relations can be used to verify object-oriented programs. The method is a natural extension of standard axiomatic techniques for specifying and verifying programs that use abstract data types.

9.7 New Replication Method

In a thesis completed in May 1989 [192], Rivka Ladin has defined a new replication method. This method is an extension of our earlier work [193] on replication techniques. Information is stored at a logically centralized service that is accessible to clients by making remote calls on its operations. To make the service highly available, so that it is likely to be accessible to clients when needed, the service is implemented by a number of replicas. Client operations take place at just one replica, and can usually be processed without delay, so the replication technique does not slow clients down. *Update* operations, which modify the state of the service, never cause a delay; the replica that performs the update communicates the new information to the other replicas in the background by using "gossip" messages. *Query* operations, which observe the state, will be delayed if an update whose effect needs to be observed is not yet known at the replica processing the operation; but this occurs rarely.

Ladin's method allows clients to indicate how operations are to be ordered. Both queries and updates can be required to occur after other updates. This is accomplished by associating each update with a unique identifier and having each query and update take a set of uids as an argument; the service will ensure that the query or update occurs after all updates whose uids are in the set.

The client-specified ordering does not provide a way to order operations that occur in parallel. To handle this case, Ladin defines two extensions that allow stronger orderings to be defined. These extensions increase the applicability of the method so that it can be used in systems where most operations are ordered by clients, but occasionally a stronger order is required.

9.8 Garbage Collection in a Distributed System

Also as part of her thesis, Ladin has defined a new technique for doing garbage collection of a distributed heap. The method uses a centralized service to keep track of inter-node references; the service is made highly available by replication, e.g., using the technique described above. The method allows nodes to garbage collect independently, using different algorithms if desired. After doing a garbage collection, a node informs the service about its references to other nodes' objects and inquires about the accessibility of any of its objects that might be accessed by other nodes. The method is thus able to detect cycles of inaccessible objects. Using the central service has several advantages: it requires fewer messages than if nodes communicate with one another directly; it offloads work from the clients to the server, thus freeing up the clients to work on behalf of their users; and it scales to large systems.

9.9 High Availability for Linda

In a thesis completed in August 1988 [313], Andrew Xu defined a method for providing high availability for the Linda tuple space [72]. The Linda tuple space is a nonstandard memory model that requires less synchronization between reads and writes than a standard model. As such, it is of interest for parallel and distributed systems because the reduced synchronization can translate into better performance. Previous implementations proposed for Linda, however, do not support high availability: if any node containing part of the tuple space fails, the memory is lost. Xu's thesis provides an efficient implementation that overcomes this problem.

9.10 Optimistic Concurrency Control in Distributed Systems

In a thesis completed in May 1989 [140], Bob Gruber extended optimistic concurrency control to work in a distributed system that supports nested atomic transactions. His thesis describes two methods. The first uses the *fixed action model* in which a transaction runs entirely at a single site and all objects that it uses are copied to that site. In the second, the *fixed object model*, objects never move; instead the transaction runs at the sites of the objects it uses. The performance of the two methods appears to be roughly the same, although the algorithms used in the fixed object model are more complicated. The fixed action model is probably the more interesting of the two because it matches recent work in distributed object-oriented data bases, in which copies of the objects used by a client reside in the client's cache.

9.11 Stable Storage Service

In a thesis finished in May 1989 [83], Jeff Cohen designed and partially implemented a new stable storage systems for Argus. The current implementation of Argus uses a disk at each node to store the stable information of all guardians at that node. This means that our stable storage is not truly stable, since a failure of a single disk can cause the loss of all stable information for that node. True stable storage would require two disks per node, and the time needed to write to stable storage would be high, since each stable write would need to be done to both disks and the two disk writes must happen sequentially [195].

To reduce these costs, Cohen worked on the new system, which is based on [92]. In his system, stable storage is provided by a stable storage service that can be accessed across the network. The service consists of three server nodes. To write or read information at the service, the Argus system at a guardian must communicate with any two of these servers. Each server has a large disk and an uninterruptible power supply. The power supply allows the server to handle a write request entirely in primary memory; the new information in the write request is written to disk later in background mode. This means that a write to an individual server takes a length of time roughly equal to a roundtrip message delay. We expect the new service to be faster than the current Argus system because the writes to the two servers can be done in parallel and the network roundtrip delay in a local area net is less than the delay for a disk write.

9.12 Publications

[1] J. Aspnes, N. Lynch, M. Merritt, A. Fekete, and W. Weihl. A theory of timestamp-based concurrency control for nested transactions. In *14th International Conference on Very Large Data Bases*, Los Angeles, CA, 1988.

[2] C. Closkey. *The Argus System Manual.* Programming Methodology Group Memo 64, MIT Laboratory for Computer Science, September 1988.

[3] A. Fekete, N. Lynch, M. Merritt, and W. Weihl. Commutativity-based locking for nested transactions. Accepted for publication, *Journal of Computer and System Sciences* (invited paper).

[4] A. Fekete, N. Lynch, M. Merritt, and W. Weihl. Commutativity-based locking for nested transactions. In *Proceedings of the Workshop on Persistent Object Systems*, Newcastle, Australia, 1989.

[5] A. Fekete, N. Lynch, M. Merritt, and W. Weihl. *Commutativity-based Locking for Nested Transactions.* Technical Memo MIT/LCS/TM-370, MIT Laboratory for Computer Science, August 1988.

[6] A. Heddaya, M. Hsu, and W. Weihl. Two phase gossip: managing distributed event histories. *Information Sciences: An International Journal*, March 1989.

[7] M. Herlihy and W. Weihl. *Hybrid Concurrency Control for Abstract Data Types.* Technical Memo MIT/LCS/TM-368, MIT Laboratory for Computer Science, August 1988.

[8] M. Herlihy, N. Lynch, M. Merritt, and W. Weihl. On the correctness of orphan elimination algorithms. *Journal of the ACM.* To appear.

[9] M. Herlihy and W. Weihl. Hybrid concurrency control for abstract data types. *Journal of Computer and System Sciences*, invited paper. To appear.

[10] E. Kolodner, B. Liskov, and W. Weihl. Atomic garbage collection: managing a stable heap. In *1989 SIGMOD Conference*, Portland, OR, 1989.

[11] E. Kolodner and W. Weihl. Crash recovery for object-oriented systems. In *1989 Workshop on Database Programming Languages*, June 1989.

[12] G. Leavens. *Verifying Object-oriented Programs that Use Subtypes.* Technical Report MIT/LCS/TR-439, MIT Laboratory for Computer Science, February 1989.

[13] B. Liskov and W. Weihl. Implementation of resilient, atomic data types. In *Readings in Object-oriented Databases*, Morgan-Kaufmann, 1988.

[14] B. Liskov. *Data Abstraction and Hierarchy.* Programming Methodology Group Memo 63, MIT Laboratory for Computer Science, July 1988.

[15] B. Liskov, L. Shrira, and J. Wroclawski. Efficient at-most-once messages based on synchronized clocks. Submitted for publication.

[16] N. Lynch, M. Merritt., W. Weihl, and A. Fekete. *A Theory of Atomic Transactions.* Technical Memo MIT/LCS/TM-362, MIT Laboratory for Computer Science, June 1988.

[17] N. Lynch, M. Merritt, W. Weihl, and A. Fekete. Atomic transactions. In progress.

[18] N. Lynch, M. Merritt, W. Weihl, and A. Fekete. A theory of atomic transactions. In *Second International Conference on Database Theory*, Bruges, Belgium, 1988.

[19] B. Oki. *Viewstamped Replication for Highly Available Distributed Systems.* Technical Report MIT/LCS/TR-423, MIT Laboratory for Computer Science, August 1988.

[20] B. Oki and B. Liskov. Viewstamped replication: a new primary copy method to support highly-available distributed systems. In *Proceedings of the Seventh ACM Symposium on Principles of Distributed Computing*, 1988.

[21] B. Oki and B. Liskov. Viewstamped replication: a new primary copy method to support highly-available distributed systems. Submitted for publication.

[22] S. Perl. *Distributed Commit Protocols for Nested Atomic Actions.* Technical Report MIT/LCS/TR-431, MIT Laboratory for Computer Science, November 1988.

[23] W. E. Weihl. Remote procedure call (chapter 3). In *Distributed Systems: An Advanced Course.* Addison-Wesley, 1989. To appear.

[24] W. Weihl. Theory of nested transactions (chapter 11). In *Distributed Systems: An Advanced Course.* Addison-Wesley, 1989. To appear.

[25] W. Weihl. Using transactions in distributed applications (chapter 10). In *Distributed Systems: An Advanced Course.* Addison-Wesley, 1989. To appear.

[26] W. Weihl. Specifications of distributed programs (chapter 19). In *Distributed Systems: An Advanced Course.* Addison-Wesley, 1989. To appear.

[27] W. Weihl. Local atomicity properties: modular concurrency control for abstract data types. *ACM TOPLAS.* To appear.

[28] W. Weihl. Linguistic support for atomic data types. *ACM TOPLAS*, to appear.

[29] W. E. Weihl. The impact of recovery on concurrency control. In *Eighth ACM Symposium on Principles of Database Systems*, Philadelphia, PA, 1989.

[30] W. Weihl. *The Impact of Recovery on Concurrency Control.* Technical Memo MIT/LCS/TM-382, MIT Laboratory for Computer Science, February 1989.

[31] W. Weihl. Commutativity-based concurrency control for abstract data types. *IEEE Transactions on Computers*, 37(12):1488–1505, December 1988.

[32] W. Weihl. Remote procedure call. In *Lecture Notes for Arctic 88: An Advanced Course on Distributed Systems.* Tromso, Norway, July 1988.

[33] W. Weihl. Using transactions in distributed applications. In *Lecture Notes for Arctic 88: An Advanced Course on Distributed Systems.* Tromso, Norway, July 1988.

[34] W. Weihl. Theory of nested transactions. In *Lecture Notes for Arctic 88: An Advanced Course on Distributed Systems.* Tromso, Norway, July 1988.

[35] W. Weihl. Specifications of distributed programs. In *Lecture Notes for Arctic 88: An Advanced Course on Distributed Systems.* Tromso, Norway, July 1988.

[36] W. Weihl. *Commutativity-based Concurrency Control for Abstract Data Types.* Technical Memo MIT/LCS/TM-367, MIT Laboratory for Computer Science, August 1988.

[37] A. Xu. *A Fault-Tolerant Network Kernel for Linda.* Technical Report MIT/LCS/TR-424, MIT Laboratory for Computer Science, August 1988.

[38] A. Xu and B. Liskov. A design for a fault-tolerant, distributed implementation of Linda. In *Proceedings of the 19th International Symposium on Fault-tolerant Computing,* IEEE, 1989. Also Programming Methodology Group Memo 65, MIT Laboratory for Computer Science, April 1989.

Theses in Progress

[1] B. Ben-Zvi. *Disconnected Actions: An Asynchronous Extension to a Nested Atomic Action System.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected 1989.

[2] S. Ghemawat. *Automatic Replication for Highly Available Services.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected 1989.

[3] S. Markowitz. *Evaluation of a Central Server Orphan Detection Scheme.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected 1989.

Theses Completed

[1] J. Cohen. *Atomic Stable Storage Across a Network.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[2] G. Curwin. *Supporting Atomicity for the Argus Object Browser.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[3] A. Farrell. *A Deadlock Detection Scheme for Argus.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, July 1988.

[4] R. Gruber. *Optimistic Concurrency Control for Nested Distributed Transactions.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[5] S. Heddaya. *Managing Event-based Replication for Abstract Data Types in Distributed Systems.* PhD thesis, Harvard University, October 1988.

[6] R. Ladin. *A Method for Constructing Highly Available Services and a Technique for Distributed Garbage Collection.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[7] G. Leavens. *Verifying Object-oriented Programs that Use Subtypes.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, December 1988.

[8] M. Levine. *Garbage Collection in Clu with Non-tagged References.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[9] C. Maeda. *Maintaining Large Knowledge Bases.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[10] A. Xu. *A Fault-Tolerant Network Kernel for Linda.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, July 1988.

Talks

[1] B. Liskov. Argus and Mercury. Lecture given at DARPA, September 1988.

[2] B. Liskov. Heterogeneous computing: a high level communication mechanism. Lecture given at National Science Foundation, December 1988.

[3] B. Liskov. Communication in the Mercury system. Lecture given at Tektronix, Inc. (August), CIPS Edmonton '88 Conference (November), 1988.

[4] B. Liskov. Communication in the Mercury system. Lecture given at Hewlett Packard Laboratories (January), Boston University (February), University of Michigan (April), Data General Corp. (April), 1989.

[5] B. Liskov. Challenges in distributed systems. Lecture given at Project MAC 25th Anniversary, October 1988.

[6] B. Liskov. Challenges in distributed systems. Lecture given at Hewlett-Packard, January 1989.

[7] B. Liskov. Programming methodology; introduction to Clu, specifying data abstractions, program construction using abstractions, using abstractions in programming languages. Lecture given at Hewlett-Packard, January 1989.

[8] B. Liskov. Introduction to Mercury. Lecture given at Siemens, February 1989.

[9] W. Weihl. Specifications of distributed programs; remote procedure call; theory of nested transactions; using transactions in distributed applications. Lecture given at Artic '88 Advanced Course on Distributed Systems, Tromso, Norway, July 1988.

[10] W. Weihl. The impact of recovery on concurrency control. Lecture given at Digital Systems Research Center (January); IBM Almaden Research Center (January); Xerox Palo Alto Research Center (January); Harvard University (March); Symposium on Principles of Database Systems (March); University of Pennsylvania (May), 1989.

Programming Systems Research

Academic Staff

D. Gifford, Group Leader

Research Staff

P. Jouvelot	M. Strauss

Graduate Students

B. Brown	J. O'Toole
M. Day	M. Reinhold
L. Lemaire	M. Sheldon

Undergraduate Students

K. Peltonen	S. Raisty
D. Segal	

Support Staff

R. Bisbee

Visitors

D. Burmaster	R. Cote

10.1 Introduction

The Programming Systems Research Group has made progress in two areas during the 1988-89 year. The group has worked on a new programming model for parallel computation involving the notion of an *effect system*. Furthermore, the group has enhanced the function of the distributed database system that we have designed and implemented.

The **prototype language FX** is a new programming model for parallel computation that combines the good features of both imperative and functional programming languages. It uses an *effect system* to investigate the use of effect specifications on controlling concurrency. An effect system is analogous to a type system (as found in many programming languages); but whereas types describe *what* results from a computation, effects classify *how* the computation proceeds. Our system deduces information that will enable the efficient parallel implementation of a broad class of polymorphic programming languages.

During the past year, our investigations with the prototype implementation have focused on:

- the design and implementation of FX subsets, one of which is used in the graduate programming language course;
- the design and testing of optimistic inference algorithms for side effect estimation of FX expressions;
- developing effect specifications for message passing concurrency and first class continuations.

We also developed **The Boston Community Information System (BCIS)** in a continuation of our work from last year. BCIS is a large scale information system that is in use at over 150 sites in the Boston area. The system was improved during the last year with the addition of an electronic mail interface to the text-based article retrieval system. The goal of our research with BCIS is to explore how the broadcast system architecture can be used to implement information systems which can support very large user populations—perhaps up to one million users.

10.2 Community Information System

During the 1988-89 year, we continued to run the Boston Community Information System experiment. This experiment provides *New York Times* and *Associated Press* news wires to our users. It consists of three main programs: a PC based version (BCIS), a TC-PIP program (Walter) and an electronic mail based system (The Clipping Service). Over the past year, over 200 Boston area homes, 40 Internet hosts and 50 electronic mail users have participated in our experiment.

The experiment also yielded exploration into other platforms. An Apple Macintosh version of the service has been prototyped and an X Windowing System implementation of the

Walter database query application has been completed. XWalter provides a "point-and-click" interface to novice users. Furthermore, Clipsend, the electronic mail portion of the experiment, is continuing to be improved to run more reliably and support a large user community. We already have users in all areas of the world, such as Japan, Switzerland, and France. We expect to be able to support well over 100 users by the end of next year.

Our experiment to charge our PC participants five dollars per month for the broadcast service has been successful; over two-thirds of the user population continues to participate. During the past year, we have spent time exploring the benefits of the technology and improving the documentation for the existing services. The experimental data report analyzed the immense amount of feedback we received from our users, and found that the Boston Community Information System provides a useful complement to existing media forms and has proved valuable to the many users in the test population.

10.3 FX Effect Analysis

The notion of effect system has been extended to deal with different aspects of compile-time analysis of programs:

- An extension to introduce the so-called "control effects" has been developed. This technique allows the introduction of first class continuations in the FX-87 programming language.
- Explicit parallelism can also be incorporated in a language that uses an effect system. A message-based extension to FX-87 has been designed and implemented on top of the experimental FX-87 Interpreter.

A major redesign of the FX programming language is under way in the PSR Group:

- A complete draft reference manual of this new version of FX has been written; Pierre Jouvelot is one of the co-editors of this specification.
- This design introduces first class modules. Among them, a vector facility inspired by Fortran 8X and the Scan Model has been proposed.

10.3.1 Reasoning about Continuations with Control Effects

First class continuations add a great deal of expressive power to a programming language as they permit the implementation of a wide variety of control structures, including jumps, error handlers, and coroutines. With this power comes substantial semantic and implementational complexities. Thus it would be very useful to be able to precisely identify which expressions in a program use first class continuations and in what manner.

We have developed a new static analysis method for first class continuations that uses an effect system to classify the control domain behavior of expressions in a a typed polymorphic language. We introduce two new control effects, **goto** and **comefrom**, that describe the control flow properties of expressions. An expression that does not have a **goto** effect is said to be continuation-following because it will always call its return continuation. An expression that does not have **comefrom** effect is said to be continuation-discarding because it will never preserve its return continuation for later use. Unobservable control effects can be masked by the effect system. Control effect soundness theorems guarantee that the effects computed statically by the effect system are a conservative approximation of the dynamic behavior of an expression.

The effect system that we describe performs certain kinds of control flow analysis that were not previously feasible. This analysis can enable a variety of compiler optimizations, including parallel expression scheduling in the presence of complex control structures. This control effect system has been implemented in the context of the FX-87 programming language.

10.3.2 Communication Effects for Message-based Concurrency

Although a fair amount of parallelism can be automatically extracted from sequential programs by smart compilers, there are some problems for which an explicitly parallel algorithm is more natural to express and easier to efficiently implement. There are numerous parallel paradigms that can be added to an otherwise sequential language to fulfill that goal, such as message passing, systolic programming, and fork/join models. We have developed a message-based communication framework based on communication effects. Communication effects are used to describe the communication behavior of expressions in a typed polymorphic programming language. Concurrency occurs between processes connected by channels on which messages are transmitted. Communication operations are characterized by two operators, out and in, depending on whether a message has been sent or received. Synchronization is only allowed by message passing along shared channels; communication via mutation of global variables is strictly prohibited by our communication effect system, thus restricting the amount of nondeterminacy in user programs.

Communication effects permit a programmer to express concurrency in a rather flexible way while preserving the correctness of implicit detection of parallelism and optimization by the compiler. This system is powerful enough to express many other parallel paradigms, like systolic arrays or pipes. This new concurrency framework has been implemented in the FX-87 programming language.

10.3.3 Polymorphism and Side Effects

We have been engaged in a project to develop a programming model for parallel computation which combines the best features of imperative and functional programming languages. The effect system which we have developed is analogous to a type system, and provides an algebraic framework for describing the behaviour of computations.

As reported last year, a prototype implementation of the FX programming language has been built. Our investigations in the area of type reconstruction systems and our performance experiments have led us to investigate several new applications of effect systems:

- a static checking system for abstract type constructors and destructors which facilitates the convenient use of modular data implementations; and
- a more flexible static type reconstruction system which may permit memory representation optimizations.

Type Reconstruction for Pattern Matching

We have developed a typing system which permits data constructor and destructor procedures to be used as first class values; this typing system permits first class procedures to double as pattern-match operators and generalizes the notion of pattern-matching.

Typechecking Polymorphic Expressions with Side Effects

We have investigated a simple typing system which associates side effects with type variables in order to permit polymorphism in the types of some expressions which perform side effects.

10.3.4 FX Large Project Programming

We have continued work on the FX module system. FX modules allow abstract types, transparent types, and values to be packaged together into first class modules. A system of *static dependent types* guarantees type safety in the presence of first class modules. We use the FX effect system to guarantee type safety in the presence of side effects. Our system, combined with a simple facility for reading (module) values from files, obviates the need for a separate linking language (as in ML or Clu).

This past year, we integrated this module system into our new prototype FX implementation. Our type reconstruction system allows many declarations within modules to be omitted and permits clients of modules to benefit from implicit polymorphism of values exported by modules.

Our implementation packages the standard types and operations into a large FX module that is available to the programmer. This allows our language design and implementation to be more modular by separating the language kernel from these standard language features. Modules also provide a convenient way to experiment with new language features as well as new versions of older features.

We are continuing to refine the module system design by investigating:

- more concise methods of combining modules to form larger ones,
- support for common idioms (like ML's **datatype**), and
- support for persistent module storage.

10.4 Publications

[1] D.K. Gifford and P. Jouvelot. Parallel functional programming: the FX project. In *International Workshop on Parallel and Distributed Algorithms*, Bonas, France, North-Holland, October 1988.

[2] R.T. Hammel and D.K. Gifford. *FX-87 Performance Measurements: Dataflow Implementation.* Technical Report MIT/LCS/TR-421, MIT Laboratory for Computer Science, November 1988.

[3] P. Jouvelot and D.K. Gifford. Reasoning about continuations with control effects. In *Proceedings of SIGPLAN '89 Conference on Programming Language Design and Implementation*, Portland, OR, June 1989.

[4] P. Jouvelot and D.K. Gifford. *Communication Effects for Message-based Concurrency.* Technical Memo MIT/LCS/TM-386, MIT Laboratory for Computer Science, 1989.

[5] P. Jouvelot and D.K. Gifford. The FX-87 interpreter. In *Proceedings of the Second IEEE International Conference on Computer Languages*, Miami Beach, FL, October 1988.

[6] P. Jouvelot and V. Dornic. FX-87, or what comes after Scheme? Special Edition of the *BIGRE Bulletin*, AFCET, France, May 1989.

[7] J.W. O'Toole, Jr. and D.K. Gifford. Type reconstruction with first class polymorphic values. In *Proceedings of the SIGPLAN '89 Conference on Programming Language Design and Implementation*, Portland, OR, June 1989.

[8] J.W. O'Toole, Jr. *A Comparison of Polymorphic Type Abstraction Rules.* Technical Memo MIT/LCS/TM-380, MIT Laboratory for Computer Science, 1988.

[9] M.A. Sheldon and D.K. Gifford. Static dependent types for first-class modules. *ACM '90 Conference on Lisp and Functional Programming.* Submitted for publication.

Theses Completed

[1] R.T. Hammel. *An FX-87 Compiler for a Dataflow Machine.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, September 1988.

[2] J. O'Toole. *Polymorphic Description Reconstruction.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[3] K. Peltonen. *A Gnu-Emacs Interface to the Community Information Systems Project.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[4] D. A. Segal. *Macnews: An Interactive News Retrieval Service for the Macintosh.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[5] M. Sheldon. *Static Dependent Types for First-class Modules.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[6] L. Zhang. *A New Architecture for Packet Switching Network Protocols.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

Theses in Progress

[1] L. LeMaire. *An Architecture for Software Reusability Tools.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected September 1990.

[2] J. O'Toole. PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected May 1992.

[3] J. Rauen. *A Module System for Scheme.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected May 1990.

[4] M. Sheldon. *An Automatically Indexed Module Repository.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected May 1992.

Talks

[1] D. Gifford. Broadcast technology for information systems. MIT ILO presentation to KDD Corporation, MIT, Cambridge, MA, October 1988.

[2] D. Gifford. Broadcast technology for information systems. MIT ILO presentation to Bell-Northern Research, MIT, Cambridge, MA, March 1989.

[3] P. Jouvelot. Pragmatics in parallel functional programming. Lecture given at Second FIRTECH Symposium, Paris, France, November 1988.

[4] P. Jouvelot. Reasoning about continuations with control effects. Lecture given at SIGPLAN '88 PLDI, Portland OR, June 1989.

[5] P. Jouvelot. Boston community information system. MIT ILO presentation to NEC, MIT, Cambridge, MA, August 1988.

[6] J. O'Toole. Type reconstruction with first class polymorphic values. Lecture given at SIGPLAN '88 PLDI, Portland, OR, June 1989.

[7] J. O'Toole. Distributed databases using broadcast media. MIT ILO presentation to Matsushita Electric Corporation, MIT, Cambridge, MA, August 1988.

[8] J. O'Toole. Modular polymorphic programming with effects. MIT ILO presentation to NEC, MIT, Cambridge, MA, April 1989.

Spoken Language Systems

Research Staff

J. Glass	S. Seneff
M. Phillips	V. Zue, Group Leader

Technical Staff

D. Goodine	K. Isaacs

Graduate Students

N. Daly	J. Pitrelli
R. Kassel	D. Rtischev
H. Leung	M. Soclof
J. Marcus	S. Trowbridge
H. Meng	

Undergraduate Students

W. Foster	C. Pao
M. McCandless	D. Whitney

Support Staff

V. Palay

Visitors

S. Ono	K. Takeda

11.1 Introduction

Spoken language input to computers is a major goal in our research in developing a graceful human-machine interface. Despite some recent successful demonstrations of speech recognition capabilities, current systems typically fall far short of human capabilities of continuous speech recognition with essentially unrestricted vocabulary and speakers, under difficult acoustic environments. Our approach to this problem is to seek a good understanding of human communication through spoken language, to capture the essential features of the process in appropriate models, and to develop the necessary computational framework to make use of these models for machine understanding.

It is our belief that the development of advanced human/machine communication systems will require expertise in signal processing, system theory, pattern recognition, and computer science. built on a solid understanding of speech science and linguistics. We place heavy emphasis on designing systems that can make use of the knowledge gained over the past four decades in human communication, with hope that such systems will one day have a performance approaching that of humans. Specifically, our approach is based on the following premises:

- The speech signal contains information regarding the intended linguistic message. It also contains information on the acoustic environment and the identity and physiological/psychological states of the speaker. As far as speech recognition is concerned, the latter sources of information can be considered as undesirable noise. Robust speech recognition is critically tied to our ability to successfully extract the linguistic information and discard those aspects that are *extra-linguistic.*

- Past research in spoken language communication has established *phonemes* as psychologically real units for representing words in the lexicon. Therefore, phonemes and other equivalent descriptors, such as distinctive features and syllables, are the most appropriate units to relate words to the speech signal for machine recognition as well.

- While phonemes are *discrete* abstract linguistic entities, their acoustic realizations in speech are inherently *continuous*, reflecting the movement of the articulators from one position to the next. Many of the acoustic cues for phonetic contrasts are encoded at specific times in the speech signal. In order to fully utilize these acoustic attributes, we believe that one must explicitly establish acoustic landmarks in the signal.

- Previous attempts at explicit utilization of speech knowledge have resulted in the development of systems that are based on heuristic rules. Such efforts typically require intense knowledge engineering, and as such are often hampered by the lack of a unified control strategy. As a result, system development is slow, and the performance fragile. In contrast, we seek to make use of the available speech knowledge by embedding such knowledge in a formal framework whereby powerful mathematical tools can be utilized to optimize its use.

- Despite significant advances made in phonetics, phonology, and other aspects of linguistics over the past decades, we still lack a complete understanding of the human speech communication process. To deal with our present state of ignorance and the inherent variability that exists throughout the process, the speech recognition system must have a stochastic component. However, it is our belief that speech-specific knowledge will enable us to build more sophisticated stochastic models than what is currently being attempted, and to reduce the amount of training data necessary for high performance.

- The ultimate goal of our research is the *understanding* of the spoken message, and the subsequent accomplishment of a task based on this understanding. To achieve this goal, we must fully integrate the speech recognition part of the problem with natural language processing so that higher level linguistic and pragmatic constraints can be utilized.

- The development of a spoken language understanding system will require interactions with several disciplines in computer science. Parallel computing will be necessary for real time processing. Efficient algorithms can greatly reduce the search space for the recognition process. Finally, theories of learning will help the system to adapt to new speakers, environments, and tasks.

The research projects in the Spoken Language Systems Group fall into several areas. First, a number of basic research topics are being explored. These include the formulation and testing of various computational models for human auditory processing, speech perception, and natural language processing, suitable for spoken language understanding. We are also attempting to quantify the acoustic cues for phonetic contrasts, and the effects of speaking rate and style on the acoustic properties of speech. Secondly, these research results are funneled into the development of an experimental spoken language system. Thirdly, alternative approaches to speech recognition, including the use of artificial neural nets and strategies derived from vision research, are being explored. Finally, part of our effort is devoted to the development of the necessary infrastructure, including the development of speech research tools and databases.

The Spoken Language Systems Group was formed in January 1989, with members drawn from the Speech Communication Group at the Research Laboratory of Electronics. While the research described herewith was conducted primarily at LCS and is intended to cover only the six months period since January, some overlap with our earlier research activities at RLE is unavoidable.

11.2 Research Reports

11.2.1 Continuous Speech Recognition: The SUMMIT System

Recently, we have put together a speech recognition system which embodies some of the research that we have been conducting in automatic speech recognition. The system, which

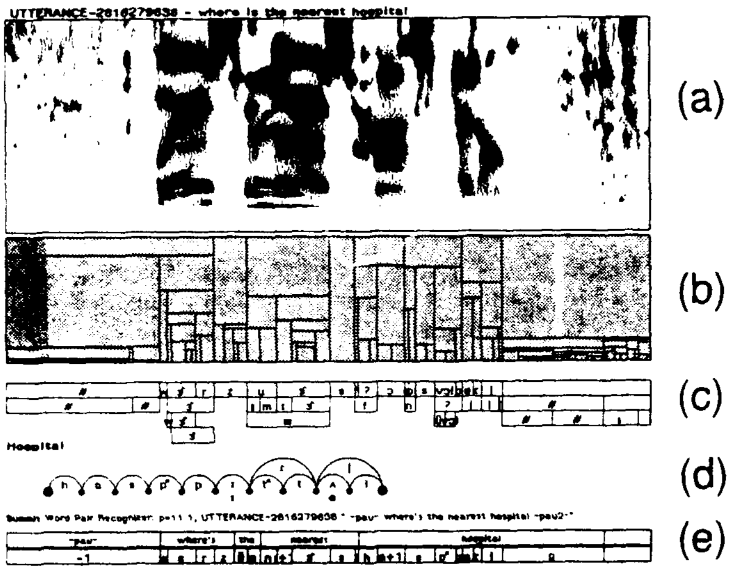

Figure 11.1: Intermediate representation leading to the recognition of the sentence, "Where is the nearest hospital?" The display contains: (a) synchrony spectrogram, (b) a dendrogram describing the multi-level acoustic segmentation, (c) a phonetic recognition network, (d) a word pronunciation network, and (e) the recognition result.

we call SUMMIT, is intended to serve as a testbed for a segmental-based approach to speech recognition. In addition, it enables us to explore how speech recognition can be integrated with natural language processing in order to achieve speech understanding.

The SUMMIT system starts the recognition process by first transforming the speech signal into a representation that models the known properties of the human auditory system [283]. The representation is illustrated in Figure 11.1(a), for the sentence "Where is the nearest hospital?" Using the output of the auditory model, acoustic landmarks of varying robustness are located and embedded in a hierarchical structure called a dendrogram [123], as shown in Figure 11.1(b). The acoustic segments in the dendrogram are then mapped to phoneme hypotheses, using a set of automatically determined acoustic parameters in conjunction with conventional pattern recognition algorithms [264]. The result is a phoneme network, in which each arc is characterized by a vector of probabilities for all the possible candidates, as shown in Figure 11.1(c).

Words in the lexicon are represented as pronunciation networks, which are generated automatically by a set of phonological rules. This is illustrated in Figure 11.1(d) for the word "hospital." Probabilities derived from training data are assigned to each arc to reflect the likelihood of a particular pronunciation. Presently, lexical decoding is accomplished by using the Viterbi algorithm to find the best path that matches the acoustic-phonetic network with the lexical network. The recognized word string is shown in Figure 11.1(e).

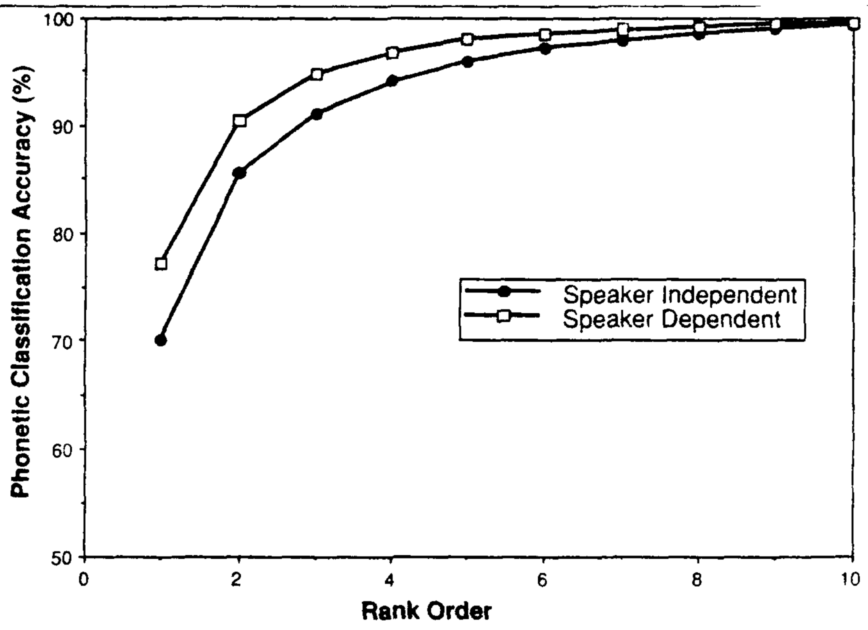

Figure 11.2: Rank order statistics for the current phone classifier on a speaker-independent task. There are 38 context-independent phone labels: 14 vowels, 3 semivowels, 3 nasals, 8 fricatives, 2 affricates, 6 stops, 1 flap, and one for silence.

We recently evaluated SUMMIT's performance in a number of ways. Phonetic classification performance was evaluated by comparing the labels provided by the classifier to those in a time-aligned transcription, using 38 context-independent phone labels [318]. This particular set was selected because it has been used in other recent evaluations within the DARPA community. For a single speaker, the top-choice classification accuracy was 77%. The correct label is within the top three nearly 95% of the time. For multiple and unknown speakers, the top-choice accuracy is about 70%, and the correct choice is within the top three over 90% of the time. Figure 11.2 shows the rank order statistics for both the speaker-dependent and speaker-independent cases.

Word accuracy for the SUMMIT system was evaluated during February on the DARPA 1000-word Resource Management task [319]. Two different speaker-independent test sets provided by NIST, consisting of 150 and 300 sentences, respectively, were used [252]. The SUMMIT system achieved a word accuracy of 87% on both test sets, using the designated word-pair grammar with perplexity of 60, and approximately 70 context-independent phone models. SUMMIT's performance compares favorably with systems that are based on hidden Markov modeling, when evaluated on the same data and using a similar number of phone models [200]. Since other researchers have been able to improve their system's performance by increasing the number of models to accommodate context-dependency, we expect that we can similarly improve SUMMIT's performance.

Currently the SUMMIT system is implemented on a Symbolics Lisp Machine augmented with an FPS Array processor, and runs in several hundred times real time. Over the next year, we will begin to port the system to a faster platform in conjunction with developed dedicated hardware to achieve near real time performance.

11.2.2 Natural Language Processing: The TINA System

A new natural language system, TINA, has been developed in our group [284] which integrates key ideas from context free grammars, Augmented Transition Networks (ATN's) [311], and Lexical Functional Grammars (LFG's) [66]. TINA is specifically designed to accommodate full integration between speech recognition and natural language processing, and has a set of features reflecting this philosophy.

The grammar begins with a set of context-free rewrite rules, which are augmented with parameters to enforce syntactic and semantic constraints. These rules are converted automatically to a network form, leading to extensive structure sharing. All arcs in the network have associated probabilities, which can be trained automatically from a set of parsed sentences. The parser uses a best-first search strategy. Control includes both top-down and bottom-up cycles, and key parameters are passed among nodes to deal with long-distance movement and agreement constraints. The probabilities provide a natural mechanism for exploring more common grammatical constructions first. TINA also includes a new strategy for dealing with movement, which can handle efficiently nested and chained gaps, and rejects crossed gaps.

Over the past few months, TINA has been ported to the DARPA 1000-word Resource Management task. We used the 791 designated training sentences and 200 (unseen) test sentences to evaluate our parser for coverage and perplexity. The training was a two-step process. We first expanded the coverage of the grammar until it could handle all of the 791 training sentences (100% coverage). We then built a new subgrammar from these sentences, with probabilities on arcs updated according to their usage within the training set (any rules that only appeared in the TIMIT domain were automatically discarded). This resulted in a grammar that was tightly defined for the RM task. We then tested this grammar for coverage and perplexity on the 200 test sentences. The results were that 84% of the test sentences were parsable, and the perplexity was 368 if all words that could follow each word were considered to be equally likely. The surprising result was that the perplexity dropped 9-fold when arc probabilities were incorporated into the measurement, down to 41.5. We also looked at the parses to establish the depth from the top of the correct parse. We found that 88% of the training sentences gave a correct parse as the first choice; this number increased to 90% for the test sentences. Both sets gave the correct parse within the top three 98% of the time.

11.2.3 Spoken Language Understanding: The VOYAGER System

Over the past three months, we initiated an effort in spoken language understanding. The project is motivated by our belief that many of the applications suitable for human/machine interaction using speech typically involve interactive problem solving. That is, in addition to converting the speech signal to text, the computer must also understand the linguistic structure of a sentence in order to generate the correct response.

In order to explore issues related to a fully-interactive spoken language system, we selected a task in which the system knows about the physical environment of a specific geographical area, and can provide assistance on how to get from one location to another within this area. The system, which we call Voyager, can also provide information concerning certain objects located inside this area. The current version of Voyager focuses on the geographic area of the city of Cambridge between MIT and Harvard University, and can answer a number of different types of questions about certain hotels, restaurants, hospitals, and other objects within this region.

Voyager is made up of three components. The first component, SUMMIT, converts the speech signal into a set of word hypotheses. The natural language component, TINA, then provides a linguistic interpretation of the set of words. The parse generated by the natural language component is then transformed into a set of query functions, which are passed to the backend for response generation. The backend is an enhanced version of the direction assistance program developed by Jim Davis of the Media Laboratory at MIT. The response generator maintains some knowledge about recent discourse history, which allows it to respond appropriately to queries such as "How do I get there?" Currently, Voyager can generate responses in the form of text, graphics, and synthetic speech.

As of now, Voyager has a vocabulary of approximately 400 words, and it can deal with about half a dozen types of queries, such as the location of objects, simple properties of objects, how to get from one place to another, and the distance and time for travel between objects. Within this limited domain of knowledge, it is our hope that Voyager will be able to handle any reasonable query that a native speaker is likely to initiate. As time progresses, Voyager's knowledge base will undoubtedly grow.

11.2.4 Isolated Word Recognition over Telephone Networks

Over the past few months, we initiated an effort to develop a small-vocabulary, isolated-word recognition system. The focus of this research is to explore how our phonetically- and segmentally-based approach will fare with the bandlimited and distorted speech transmitted through local and long distance telephone networks, spoken by real users.

As a first step, we selected the task of recognizing a small set of city names. We have implemented such a system, and have begun some preliminary evaluations, using data collected by NYNEX Corporation. We are also using this simple task as a framework in which to explore the use of unsupervised learning techniques to enable the automatic expansion of the vocabulary.

11.3 Student Reports

Nancy Daly

During the spring semester, Daly spent most of her time working as a teaching assistant for a new course on automatic speech recognition introduced by Victor Zue. Over the next few months, she plans to take her area exam and work on her doctoral thesis, which is in the area of prosodic aids for speech recognition.

Prosody is the stress, rhythm, and intonation of speech. While the importance of prosodic information has long been documented for human speech communication, automatic speech recognition systems developed as of now have all but ignored this source of information. The purpose of her thesis research is to see how prosodic information could be incorporated into speech recognition systems to improve their performance.

One of the first projects is to determine what form of prosodic information can be reliably extracted from the speech signal in the absence of any segmental information. Specifically, she is investigating whether stressed syllables can be identified reliably by native listeners when the phoneme identity has been removed from the speech signal through inverse filtering. This line of investigation can lead to the determination of the stress pattern of words, and the use of this knowledge to aid phonetic recognition and lexical access.

Rob Kassel

Kassel is pursuing a Master's thesis on the use of distinctive features for lexical access. Distinctive features have been proposed by many as a sub-phoneme linguistic unit. Feature spreading can concisely represent the allophonic variation found in spoken language, an attractive property for speech recognition systems. He began a study to determine the expressive power of distinctive features in terms of information theoretical measures.

Hong Leung

Leung just completed his Ph.D. thesis entitled "The Use of Artificial Neural Networks for Phonetic Recognition." One of the major problems with current speech recognition systems is that the system's self-organizing framework is very powerful but too rigid for incorporating more human knowledge about speech, or that there is a significant amount of human knowledge in the system but the control strategy is not powerful enough. Due to their flexible self-organizing framework, artificial neural networks (ANN's) can potentially bridge the gap between our knowledge in speech and powerful self-organizing mechanisms. Leung's thesis is concerned with the use of ANN's for phonetic recognition. There are three major objectives. First, by investigating ANN's in order to gain a better understanding of their basic characteristics and capabilities, we may be able to exploit them more fully as pattern classifiers. Secondly, by properly applying our acoustic-phonetic knowledge, we can potentially enhance the flexible framework of ANN's for phonetic recognition. Thirdly, by comparing them with traditional pattern classification techniques, we can better understand the merits and shortcomings of the different approaches.

The multi-layer perceptron (MLP) was selected for his investigation, which centered around a set of vowel recognition experiments. In order to isolate different sources of variability in the speech signal, four different databases were used for our study. The largest database consists of 22,000 vowel tokens extracted from continuous sentences in the TIMIT database, spoken by 550 male and female speakers. The performance of the network was evaluated in several ways. Evaluation in terms of average agreement with the phonetic transcription suggests that the performance of the network compares favorably to human performance in perceptual experiments. Evaluation along the phonological dimension suggests that most of the confusions between the network and transcription labels are quite reasonable.

Next, the characteristics and representations of the MLP were explored. Specifically, he examined the performance of the network as a function of the number of training iterations, amount of training data, number of hidden units, number of hidden layers, and use of the nonlinear sigmoid function. He also discussed the structure and self-organization of the internal representations, choices for output representations, and the use of heterogeneous input representations. Other issues discussed include error metrics for training the network, initializations of the network, and rapid adaptation of the network to a new speaker.

Finally, the performance of the network was compared with that of two traditional classification techniques. For the vowel classification task, experiments demonstrate that the MLP can yield higher performance than k-nearest neighbor and Gaussian classifiers. The results suggest that the MLP can provide an effective alternative for pattern classification, especially if the classification problem is not well understood.

Jeffrey Marcus

Marcus has been working on incorporating speech units of different sizes (e.g., phoneme, diphone, word) in a speech recognition system. In addition, he is considering schemes for sharing information among speech units which have certain phonetic similarities so that parameter estimates for these models are improved. Another major goal of his work is to advance recognizer design methodology by demonstrating the utility of statistical and data analytic techniques which have not been applied previously.

The work is currently focused on modeling function words such as "the" and "and," since they vary greatly acoustically and cause a disproportionate number of recognizer errors. In the future, these techniques will be extended to other lexical and phonetic units.

Helen Meng

Meng joined the group in January, and spent the past semester finishing her Bachelor's thesis, and building up background in speech through taking the courses 6.979, *Automatic Speech Recognition* and 6.541J, *Speech Communication.* In addition, she attended spectrogram reading sessions run in the Spoken Language Systems Group. A term paper was also written under the topic of "An Acoustic Study of the Semi-Vowel /l/." The paper reports a study of prevocalic, intervocalic and postvocalic /l/'s in some data collected by Dennis Klatt. Over the next few months, she will be familiarizing herself with the computational facilities in the group, as well as searching for a topic for her Master's thesis.

John F. Pitrelli

Pitrelli has been studying phoneme durations in order to develop a duration model to aid speech recognition. Duration is potentially a strong cue for certain phonemic distinctions, including inherently long vs. short vowels, and voiced vs. unvoiced obstruent consonants. Phoneme durations are affected, though, by an abundance of factors ranging from detailed phonetic context effects to syntax and semantics. Our lack of understanding of these effects and their interactions hinders our use of potentially useful duration information to the extent that most speech recognition systems currently use only rudimentary duration models or use time-warping procedures, which distort duration information.

Recent research has focused on two facets of the duration modeling problem. One is the completion and evaluation of a hierarchical model accounting for discrete-valued factor variables, such as phonetic context and syntactic-unit-final lengthening. The other task has been a preliminary exploration of the effects of speaking-rate variations on phoneme duration. Future work includes the continuation of the speaking-rate experiments, with the goal of improving understanding of rate effects on duration, both gradual, such as vowel compression, and abrupt, such as flapping of alveolar stops. Following these experiments, the hierarchical duration model will be augmented by the incorporation of an appropriate function of speaking rate.

Dimitry Rtischev

Rtischev joined the group in January. Over the past five months, he worked on an interactive software facility for simulation of hidden Markov models. The completed program, named HIMARK, provides a flexible experimental environment for constructing, training, and observing hidden Markov models and using them for various speech recognition tasks. HIMARK formed the basis for two lab assignments which he prepared for 6.979, *Automatic Speech Recognition.* Dimitry's plans for the next year include research in applying statistical methods such as HMM for speech synthesis and preparing for the Preliminary Written Examination and Oral Exam.

Michal Soclof

Soclof joined the group in January, and spent the spring semester learning about automatic speech recognition by taking the speech recognition and spectrogram reading courses, and by reading relevant material. In addition, she learned about the SUMMIT system and became familiar with the computational facilities in the group. During the upcoming months, she will be working on her Master's thesis research. A potential topic which she is investigating is the problem of detecting speech in the presence of other vocalizations. This entails being able to distinguish between a speech event and a non-speech event such as throat clearing or coughing. She will be studying what makes the two events different and possible methods for distinguishing them.

Sean Trowbridge

Trowbridge joined the group in January, and spent the spring term mainly getting oriented, learning about speech recognition in general, and spectrogram reading in particular. He also helped out with some grading for 6.979, and started preliminary research for his Master's thesis. In the coming year, he will be working with Steve Ward on a thesis that involves the NuMesh computer and its application to speech recognition. He will be designing something resembling a compiler for the machine, which will take a computation specification, along with some resource constraints, and produce a topology and timing for each processor that will perform the computation specified.

11.4 Publications

[1] J. Glass, M. Phillips, S. Seneff, and V. Zue. Acoustic segmentation and phonetic classification in the SUMMIT system. In *Proceedings from ICASSP*, pages 389–392, Glasgow, Scotland, 1989.

[2] J. Glass, M. Phillips, S. Seneff, and V. Zue. The MIT SUMMIT speech recognition system: a progress report. In *Proceedings from Speech and Natural Language Workshop*, pages 179–189, Philadelphia, PA, 1989.

[3] S. Seneff. TINA: a probabilistic syntactic parser for speech understanding systems. In *Proceedings from Speech and Natural Language Workshop*, pages 168–178, Philadelphia, PA, 1989.

Thesis Completed

[1] H.C. Leung. *The Use of Artificial Neural Nets for Phonetic Recognition.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, May 1989. Supervised by V.W. Zue.

Theses in Progress

[1] N. Daly. *Prosodic Aids to Speech Recognition.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected 1991. Supervised by V.W. Zue.

[2] R. Kassel. *The Information Content of Distinctive Features in American English.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected 1989. Supervised by V.W. Zue.

[3] J. Marcus. *Incorporation of Different Sized Units in a Speech Recognition System.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected 1991. Supervised by V.W. Zue.

[4] J. Pitrelli. *Duration Models for Continuous Speech Recognition.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected December 1989. Supervised by V.W. Zue.

Talks

[1] V.W. Zue. Speech recognition at MIT. Lecture given at the External Research Symposium, Apple Computer, Inc. Cupertino, CA, January 1989.

[2] V.W. Zue. Speech recognition at MIT. Lecture given at Siemens A.G., Munich, West Germany, February 1989.

[3] S. Seneff. TINA: a probabilistic syntactic parser for speech understanding systems. Lecture given at AAAI Workshop on Spoken Language Systems, Stanford University, March 1989.

Systematic Program Development

Academic Staff

J.V. Guttag, Group Leader

Research Staff

S.J. Garland

Graduate Students

D. N. Jackson	M. T. Vandevoorde
M. Reinhold	K. A. Yelick
W. Shang	

Undergraduate Students

J. Chen	J. Rauen
M. Lillibridge	

Support Staff

A. Rubin

Visitors

T. Nipkow M.C. Godfrey

12.1 Introduction

The Systematic Program Development, though small, has a diverse set of interests. These include programming methodology, programming multiprocessors, specification languages (in conjunction with researchers at the Digital Equipment Corporation), circuit verification (in conjunction with researchers at the Technical University of Denmark), automatic theorem proving, and high performance garbage collection.

12.2 The Larch Family of Specification Languages

The Larch family of specification languages supports a two-tiered definitional approach to specification. Each specification has components written in two languages: one designed for a specific programming language and another independent of any programming language. The former are called *Larch interface languages*, and the latter the *Larch Shared Language* (*LSL*).

Larch interface languages are used to specify the interfaces between program components. Each specification provides the information needed to use the interface and to write programs that implement it. A critical part of each interface is how the component communicates with its environment. Communication mechanisms differ from programming language to programming language, sometimes in subtle ways. We have found it easier to be precise about communication when the interface specification language reflects the programming language. Specifications written in such interface languages are generally shorter than those written in a "universal" interface language. They are also clearer to programmers who implement components and to programmers who use them.

Each Larch interface language deals with what can be observed about the behavior of components written in a particular programming language. It incorporates programming-language-specific notations for features such as side effects, exception handling, iterators, and concurrency. Its simplicity or complexity depends largely upon the simplicity or complexity of the observable state and state transformations of its programming language. Figure 12.1 contains a sample interface specification for a CLU procedure in a window system.

Larch Shared Language specifications are used to provide a semantics for the primitive terms used in interface specifications. Specifiers are not limited to a fixed set of primitive terms, but

$addWindow =$ **proc** $(v : View, w : Window, c : Coord)$ **signals** $(duplicate)$
  modifies v
  ensures $v' = addW(v, w, c)$
    except when $w \in v$ **signals** *duplicate* **ensures** $v' = v$

Figure 12.1: Sample Larch/CLU Interface Specification

can use LSL to define specialized vocabularies suitable for particular interface specifications. For example, an LSL specification would be used to define the meaning of the symbols $\in$ and *addW* in Figure 12.1, thereby precisely answering questions such as what it means for a window to be in a view (visible or possibly obscured?), or what it means to add a window to a view that may contain other windows at the same location.

The Larch approach encourages specifiers to keep most of the complexity of specifications in the LSL tier for several reasons:

- LSL abstractions are more likely to be re-usable than interface specifications.
- LSL has a simpler underlying semantics than most programming languages (and hence than most interface languages), so that specifiers are less likely to make mistakes.
- It is easier to make and check claims about semantic properties of LSL specifications than about semantic properties of interface specifications.

12.3 The LP Theorem Proving System

LP has changed dramatically in the last year. We take this opportunity to present a fairly detailed overview of its current capabilities.

The basis for proofs in LP is a logical system consisting of equations, rewrite rules, operator theories, induction rules, and deduction rules, all expressed in a multisorted fragment of first-order logic. A logical system in LP is closely related to an LSL theory, but is handled in somewhat different ways, both because axioms in LP have operational content as well as semantic content and because they can be presented to LP incrementally, rather than all at once.

12.3.1 Declarations

Sorts, operators, and variables play exactly the same roles in LP as they do in LSL. They must be declared, and operators can be overloaded. The syntax for operators at the moment is not as rich as in LSL, but we plan to rectify that. Unlike LSL, LP at present provides no scoping for variables.

12.3.2 Equations and Rewrite Rules

LP is based on a fragment of first-order logic in which equations play a prominent role. Some of LP's inference mechanisms work directly with equations. Most, however, require that equations be oriented into rewrite rules, which LP uses to reduce terms to normal forms. It is usually essential that the rewriting relation be terminating, i.e., that no term can be rewritten infinitely many times. LP provides several mechanisms that automatically orient many sets of equations into terminating rewriting systems. For example, in response to the commands

> **declare sort** G
> **declare variables** x, y, z: G
> **declare operators** e: $\rightarrow G$, i: $G \rightarrow G$, $\_ * \_$: $G, G \rightarrow G$
> **assert**
> $(x * y) * z == x * (y * z)$
> $e == x * i(x)$
> $e * x == x$
> ..

that enter the usual first-order axioms for groups, LP produces the rewrite rules

$$\begin{array}{l} (x * y) * z \rightarrow x * (y * z) \\ x * i(x) \rightarrow e \\ e * x \rightarrow x. \end{array}$$

It automatically reverses the second equation to prevent nonterminating rewriting sequences such as $e \rightarrow e * i(e) \rightarrow i(e) \rightarrow i(e * i(e)) \rightarrow i(i(e)) \rightarrow \ldots$ The discussion of operator theories, below, treats the issue of termination further.

A system's rewriting theory (i.e., the propositions that can be proved by reduction to normal form) is always a subset of its equational theory (i.e., the propositions that follow logically from its equations and from its rewrite rules considered as equations). The proof mechanisms discussed below compensate for the incompleteness that results when, as is usually the case, a system's rewriting theory does not include all of its equational theory. In the case of group theory, for example, the equation $e == i(e)$ follows logically from the second and third axioms, but is not in the rewriting theory of the three rewrite rules (because it is irreducible and yet is not an identity).

LP provides builtin rewrite rules to simplify terms involving the Boolean operators $\neg$, $\&$, $|$, $\Rightarrow$, and $\Leftrightarrow$, the equality operator $=$, and the conditional operator **if**. These rewrite rules are sufficient to prove many, but not all, identities involving these operators. Unfortunately, the sets of rewrite rules that are known to be complete for propositional calculus require exponential time and space. Furthermore, they can expand, rather than simplify, propositions that do not reduce to identities. These are serious drawbacks, because when we are debugging specifications we often attempt to prove conjectures that are not true. So none of the complete sets of rewrite rules is built into LP. Instead, LP provides proof mechanisms that can be used to overcome incompleteness in a rewriting system, and it allows users to add any of the complete sets they choose to use.

LP treats the equations *true* $==$ *false* and $x == t$, where t is a term not containing the variable x, as inconsistent. Inconsistencies can be used to establish subgoals in proofs by cases and contradiction. If they arise in other situations, they indicate that the axioms in the logical system are inconsistent.

12.3.3 Operator Theories

LP provides special mechanisms for handling equations such as $x + y == y + x$ that cannot be oriented into terminating rewrite rules. The LP command **assert ac** + says that + is associative and commutative. Logically, this assertion is merely an abbreviation for two equations. Operationally, LP uses it to match and unify terms modulo associativity and commutativity. This not only increases the number of theories that LP can reason about, but also reduces the number of axioms required to describe various theories, the number of reductions necessary to derive identities, and the need for certain kinds of user interaction, e.g., case analysis. The main drawback of term rewriting modulo operator theories is that it can be much slower than conventional term rewriting.

LP recognizes two nonempty operator theories: the associative-commutative theory and the commutative theory. It contains a mechanism (based on user-supplied polynomial interpretations of operators) for ordering equations that contain commutative and associative-commutative operators into terminating systems of rewrite rules. But this mechanism is difficult to use, and most users rely on simpler ordering methods based on LP-suggested partial orderings of operators. These simpler ordering methods do not guarantee termination when equations contain commutative or associative-commutative operators, but they work well in practice. Like manual ordering methods, which give users complete control over whether equations are ordered from left to right or from right to left, they are easy to use. In striking contrast to manual ordering methods, they have not yet caused difficulties by producing a nonterminating set of rewrite rules.

12.3.4 Induction Rules

LP uses induction rules to generate subgoals to be proved for the basis and induction steps in proofs by induction. The syntax for induction rules is the same in LP as in LSL.[1] Users can specify multiple induction rules for a single sort, e.g., by the LP commands

declare sorts E, S
declare operators
   $\{\}\colon \rightarrow S$
   $\{\_\_\}\colon E \rightarrow S$
   $\_\_ \cup \_\_\colon S, S \rightarrow S$
   $insert\colon S, E \rightarrow S$
   ..
set name *setInduction1*
assert S **generated by** {}, *insert*
set name *setInduction2*
assert S **generated by** {}, {__}, ∪

and can use the appropriate rule when attempting to prove an equation by induction; e.g.,

[1]The semantics of induction is stronger in LSL than in LP, where arbitrary first-order formulas cannot be written.

prove $x \subseteq (x \cup y)$ **by induction on** x **using** *setInduction2*

In LSL, the axioms of a trait typically have only one **generated by** for a sort. It is often useful, however, to put others in the trait's implications.

12.3.5 Deduction Rules

LP subsumes the logical power of the **partitioned by** construct of LSL by allowing users to assert deduction rules, which LP uses to deduce equations from other equations and rewrite rules. In general, a **partitioned by** is equivalent to a universal existential axiom, which can be expressed as a deduction rule in LP. For example, the LP commands

declare sorts E, S
declare operator $\in$: $E, S \rightarrow$ *Bool*
declare variables e: E, x, y: S
assert when (forall e) $e \in x == e \in y$ **yield** $x == y$

define a deduction rule equivalent to the axiom

$$(\forall x, y : S)\,[(\forall e : E)(e \in x \Leftrightarrow e \in y) \Rightarrow x = y]$$

of set extensionality, which can also be expressed by **assert** S **partitioned by** $\in$ in LP, as in LSL. This deduction rule enables LP to deduce equations such as $x == x \cup x$ automatically from equations such as $e \in x == e \in (x \cup x)$.

Deduction rules also serve to improve the performance of LP and to reduce the need for user interaction. Examples of such deduction rules are the builtin &-splitting law

declare variables p, q: *Bool*
when $p \,\&\, q == \mathit{true}$ **yield** $p == \mathit{true}$, $q == \mathit{true}$

and the cancellation law for addition

declare variables x, y, z: *Nat*
when $x + y == x + z$ **yield** $y == z$

LP automatically applies deduction rules to equations and rewrite rules whenever they are normalized.

12.3.6 Proof Mechanisms in LP

This section provides a brief overview of the proof mechanisms in LP.

LP provides mechanisms for proving theorems using both forward and backward inference. Forward inferences produce consequences from a logical system; backward inferences produce lemmas whose proof will suffice to establish a conjecture. There are four methods of forward inference in LP.

- Automatic *normalization* produces new consequences when a rewrite rule is added to a system. LP keeps rewrite rules, equations, and deduction rules in normal form. If an equation or rewrite rule normalizes to an identity, it is discarded. If the hypothesis of a deduction rule normalizes to an identity, the deduction rule is replaced by the equations in its conclusions. Users can "immunize" equations, rewrite rules, and deduction rules to protect them from automatic normalization, both to enhance the performance of LP and to preserve a particular form for use in a proof. Users can also "deactivate" rewrite rules and deduction rules to prevent them from being automatically applied.

- Automatic *application of deduction rules* produces new consequences after equations and rewrite rules in a system are normalized. Deduction rules can also be applied explicitly, e.g., to immune equations.

- The computation of *critical pairs* and the Knuth-Bendix *completion* procedure produce consequences (such as $i(e) == e$) from incomplete rewriting systems (such as the three rewrite rules for groups). We rarely complete our rewriting systems. However, we often make selective use of critical pairs. We also use the completion procedure to look for inconsistencies.

- Explicit *instantiation* of variables in equations, rewrite rules, and deduction rules also produces consequences. For example, in a system that contains the rewrite rules $a < (b + c) \rightarrow \mathit{true}$ and $(b + c) < d \rightarrow \mathit{true}$, instantiating the deduction rule

  $$\textbf{when } x < y == \mathit{true},\ y < z == \mathit{true} \textbf{ yield } x < z == \mathit{true}$$

  with a for x, $b + c$ for y, and d for z produces a deduction rule whose hypotheses normalize to identities, thereby yielding the conclusion $a < d \rightarrow \mathit{true}$.

There are also six methods of backward inference for proving equations in LP. These methods are invoked by the **prove** command. In each method, LP generates a set of subgoals, i.e., lemmas to be proved that together are sufficient to imply the conjecture. For some methods, it also generates additional axioms that may be used to prove particular subgoals.

- *Normalization* rewrites conjectures. If a conjecture normalizes to an identity, it is a theorem. Otherwise the normalized conjecture becomes the subgoal to be proved.

- *Proofs by cases* can further rewrite a conjecture. The command **prove** e **by cases** $t_1, \ldots, t_n$ directs LP to prove an equation e by division into cases $t_1, \ldots, t_n$ (or into two cases, t_1 and $\neg(t_1)$, if $n = 1$). One subgoal is to prove $t_1 \mid \ldots \mid t_n$. In addition, for each i from 1 to n, LP substitutes new constants for the variables of t_i in both t_i and e to form t_i' and e_i', and it creates a subgoal e_i' with the additional hypothesis $t_i' \rightarrow \mathit{true}$. If an inconsistency results from adding the case hypothesis t_i', that case is impossible, so e_i' is vacuously true.

  Case analysis has two primary uses. If the conjecture is a theorem, a proof by cases may circumvent a lack of completeness in the rewrite rules. If the conjecture is not a theorem, an attempted proof by cases may simplify the conjecture and make it easier to understand why the proof is not succeeding.

- *Proofs by induction* are based on the induction rules described above.

- *Proofs by contradiction* provide an indirect method of proof. If an inconsistency follows from adding the negation of the conjecture to LP's logical system, then the conjecture is a theorem.

- *Proofs of implications* can be carried out using a simplified proof by cases. The command **prove** $t_1 \Rightarrow t_2$ **by** $\Rightarrow$ directs LP to prove the subgoal t_2' using the hypothesis $t_1' \rightarrow \mathit{true}$, where t_1' and t_2' are obtained as in a proof by cases. (This suffices because the implication is vacuously true when t_1' is false.)

- *Proofs of conjunctions* provide a way to reduce the expense of equational term rewriting. The command **prove** $t_1 \,\&\, \ldots \,\&\, t_n$ **by** & directs LP to prove $t_1, \ldots, t_n$ as subgoals.

LP allows users to determine which of these methods of backward inference are applied automatically and in what order. The LP command

set proof-method & , $\Rightarrow$, **normalization**

directs LP to use the first of the three named methods that applies to a given conjecture.

Proofs of interesting conjectures hardly ever succeed on the first try. Sometimes the conjecture is wrong. Sometimes the formalization is incorrect or incomplete. Sometimes the proof strategy is flawed or not detailed enough. When an attempted proof fails, we use a variety of LP facilities (e.g., case analysis) to try to understand the problem. Because many proof attempts fail, LP is designed to fail relatively quickly and to provide useful information when it does. It is not designed to find difficult proofs automatically. Unlike the Boyer-Moore prover, it does not perform heuristic searches for a proof. Unlike LCF, it does not allow users to define complicated search tactics. Strategic decisions, such as when to try induction, must be supplied as explicit LP commands (either by the user or by a front-end such as LSLC). On the other hand, LP is more than a "proof checker," since it does not require proofs to be described in minute detail. In many respects, LP is best described as a "proof debugger."

12.4 Hardware Verification

For many years, engineers have used simulation to convince themselves that the circuits they design behave as intended. As circuits get more complex, it becomes tempting to augment simulation with formal proofs. Typically, these proofs involve a large number of simple steps. Doing them by hand is cumbersome, boring, and prone to mistakes. Unless these proofs are machine generated or machine checked, there is very little reason to believe them.

In the past year, we have conducted several successful experiments using a theorem prover, LP, to verify properties of VLSI circuits. We started with several circuits that had previously been verified by hand. We then tried to construct machine checked proofs with the same structure as the original proofs. In the process of using LP to verify the circuits, we uncovered several minor errors in, and simplifications to, the original circuits and manual proofs.

Any formalized verification of a circuit must be based on an abstract description of the circuit. The choice of descriptive mechanism depends upon the intended use. Differential equations, for example, are useful in verifying physical properties such as power consumption, timing, or heat dissipation. Our approach is aimed at verifying functional properties of a design, and is based on describing the circuit as a parallel program, using a language, Synchronized Transitions, developed by Jørgen Staunstrup of the Technical University of Denmark.

While the circuits we have verified are not particularly complex, our experiments yielded several interesting insights. These include:

- Even for simple circuits, one cannot rely on proofs that have not been machine checked.
- Combined with Synchronized Transitions, the technique of invariant assertions used to verify safety properties of concurrent programs is useful for machine-aided reasoning about circuits.
- The verification process is quite sensitive to the exact way in which the problem is formulated. For example, proofs seem to work better when induction can be done over the structure of the circuit rather than over time.
- Circuit verification seems more amenable to machine checking than traditional program verification.
- The style of mechanical theorem proving supported by LP seems well suited to reasoning about circuits.

12.5 Programming Multiprocessors

In this research, we consider the problem of writing explicitly parallel programs for small to medium sized multiprocessors. To limit the scope of the work, the target applications are assumed to be symbolic problems, which are characterized by data structures that are irregular and dynamic and by a low percentage of numerical operations. In addition, we consider only applications for which a sequential solution is possible, i.e., concurrency is used to improve performance but is not inherent in the problem definition.

Various methods have been proposed to address the problem of writing and reasoning about parallel programs, but these tend to address only single module programs. Notions of module composition are missing and as a result, a style of program development is encouraged in which the entire program is designed and implemented as a single unit. Conversely, methods that have been developed for "programming in the large" of sequential programs are not applicable, and attempts to extend them to parallel programs often result in programs that exhibit very little real concurrency. Our goal is to be able to decompose parallel programs into independently specifiable units without prohibiting efficient implementations.

The research is organized around an extended example application, which involves parallel algorithm synthesis, correctness arguments, program module specifications, an implementation, and performance measurements. The application is to solve the completion problem for term rewriting systems, for which the well known Knuth and Bendix procedure is a sequential solution. Although the completion problem has been studied extensively, there are currently no parallel solutions.

Our parallel solution can be abstractly described by a set of inference rules that are nondeterministically applied to a system of rewrite rules and equations. In recent years, there has been a trend toward describing sequential completion procedures in this manner, often leading to better algorithms and simpler correctness arguments. The framework is especially useful in the context of concurrency, since there are known techniques for reasoning about concurrent systems using the possible sequences of state transitions. In this respect, our set of inference rules defines the set of possible state transitions, but there is an important difference in how we intend to reason about these systems. Rather than reasoning about the behavior of each high level transition by considering directly the sequence of low level operations that implement it, we intend to use the specifications of the underlying objects and their operations; each object is thus an independent unit which can be reused in any other context for which the same specification is required.

The implementation effort is still in progress but has already uncovered a number of interesting tradeoffs in the general programming problem. For example, concurrent data types that present the illusion of sequential access often lead to poor performance. The performance may be characterized by either long latency of operations, or little actual concurrency of multiple operations. In addition, examples of highly concurrent data types tend to have complex, almost mysterious implementations. In some cases, specification detail can be added to allow implementations that are more efficient or less complex, but for this we pay a price in terms of generality of the abstraction. We currently have an implementation of a completion procedure that exploits a small amount of parallelism, and closely mimics the behavior of a sequential implementation. As we add more parallelism to the procedure, the mapping between the parallel and sequential solutions will become less obvious, and the correctness argument more difficult.

12.6 Logic Programming

Most of the theoretical work on the semantics of logic programs assumes an interpreter that provides a complete resolution procedure. In contrast, for reasons of efficiency, most logic programming languages are built around incomplete procedures. This difference is rooted in Prolog, which evaluates resolvent trees in a depth-first rather than a breadth-first order. The gap is widened by some equational logic languages, which combine the incompleteness of depth-first evaluation with incomplete approximations to equational unification. Because of this gap, it is unsound to reason about logic programs using their declarative semantics. This in turn makes it difficult to develop abstraction mechanisms that can be used to partition a logic program into independently specifiable modules.

In this work, we considered the role type systems can play in closing the gap between the operational and declarative semantics of logic programs. We develop the notion of an equational mode system for use in constraining the domains of both predicates and unification procedures. The mode system is used to guide the resolution-based interpreter, and as a result, we can show that two predicate implementations with the same declarative meaning will be operationally equivalent.

This work was done in conjunction with Joseph Zachary of the University of Utah.

12.7 Publications

[1] S.J. Garland, J.V. Guttag, and J.S. Staunstrup. Verification of VLSI circuits using LP. In *Proceedings of International Workshop on Design for Behavioral Verification*, Glasgow, Scotland, July 1988.

[2] S.J. Garland and J.V. Guttag. An overview of LP, the Larch prover. In *Third International Conference on Rewriting Techniques and Applications*, Chapel Hill, NC, May 1989.

[3] S.J. Garland, J.V. Guttag, and J.S. Staunstrup. Compositional verificaiion of VLSI circuits. In *Workshop on Automatic Verification Methods for Finite State Systems*, Grenoble, France, June 1989.

[4] M.B. Reinhold. *Type Checking is Undecidable When "Type" is a Type*, Technical Report MIT/LCS/TR-458, MIT Laboratory for Computer Science, December 1989.

[5] M.T. Vandevoorde and E.S. Roberts. WorkCrews: an abstraction for controlling parallelism. *International Journal of Parallel Programming*, 17(4), August 1988.

[6] K.A. Yelick and J.L. Zachary. Moded type systems for logic programming. In *Proceedings of Principles of Programming Languages Conference*, January 1989.

[7] P. Zave and D. Jackson. Practical specification techniques for control-oriented systems. In *Proceedings of the IFIP 11th World Computer Congress*, San Francisco, CA, 1989.

Theory of Computation

Academic Staff

B. Awerbuch	C. Leiserson	R. Rivest, Group Leader
P. Elias	N. Lynch	D. Shmoys
S. Goldwasser	A. Meyer	M. Sipser
F. Leighton	S. Micali	É. Tardos

Graduate Students

R. Ashcroft	W. Goddard	R. Koch	A. Rudich
J. Aslam	S. Goldman	D. Kravets	R. Schapire
M. Bellare	K. Graves	B. Maggs	E. Schwabe
B. Berger	R. Greenberg	S. Malitz	L. Schulman
B. Bloom	M. Grigni	Y. Mansour	J. Siskind
A. Blum	C. Haibt	S. Mentzer	R. Sloan
T. Cormen	M. Hansen	M. Newman	M. Soclof
L. Cowen	R. Hirschfeld	M. Papaefthymiou	C. Stein
C. Crépeau	A. Ishii	J. Park	M. Tuttle
C. Chan	L. Jategaonkar	C. Phillips	P. Wang
A. Dhagat	T. Jim	S. Plotkin	J. Wein
R. Ehrenborg	B. Kaliski	S. Rao	S.-M. Wu
B. Eisenberg	J. Kilian	J. Riecke	J. Yang
L. Fortnow	S. Kipnis	P. Rogaway	
J. Fried	P. Klein	J. Rompel	

Undergraduate Students

M. Ernst	R. Kaseguma	P. Wang
J. Fernandez	S. Trowbridge	D. Williamson

Support Staff

S. Bemus	D. Crowell	B. Hubbard
C. Brownlie	D. Grupp	S. Merritt

Visitors and PostDocs

W. Aiello	M. Kearns	S. Safra
S. Cosmadakis	L. Levin	N. Shavit
S. Istrail	N. Nisan	A. Sherman

13.1 Introduction

The MIT/LCS Theory of Computation (TOC) Group is one of the largest theoretical computer science research groups in the world. It includes faculty, students, and visitors from both the Departments of Electrical Engineering and Computer Science, and Applied Mathematics.

The principal **research areas** investigated by members of the TOC Group are:

- algorithms: combinatorial, geometric, graph-theoretic, number theoretic;
- cryptology;
- computational complexity;
- parallel computation;
- distributed computation: algorithms and semantics;
- machine learning;
- semantics and logic of programs; and
- VLSI design theory.

Group members were responsible for over 150 publications and several dozen public lectures around the world during the past year. The *individual reports* by faculty and students in the next sections, and the *annotated reference and lecture lists* offer further descriptions of the year's activities.

The following **major research contributions** merit highlighting:

- Awerbuch, Mansour, and Shavit's polynomial solution to the basic network problem of "end to end communication".
- Awerbuch and Sipser's efficient implementation (constant time overhead) of the new notion of a "synchronizer for dynamic networks" implying that dynamic networks are as fast as static networks.
- Elias' geometric demonstration that reliable communication at a positive rate is possible over a channel which introduces a fraction $1/2 - \epsilon$ of errors, so long as the receiver is allowed to list $O(1/\epsilon^2)$ possible transmitted codewords rather than just one.
- Fortnow and Sipser's oracle collapsing the probabilistic polynomial time hierarchy.

- Koch proved a decade old conjecture about the expected throughput of the dilated butterfly switching network that has application to the optimal design of networks like that used in the BBN Butterfly Machine (Ph.D. thesis).
- Leighton and Maggs developed highly efficient packet routing algorithms for a twin butterfly. The algorithms are the first fault-tolerant routing algorithms for bounded degree switching networks, and appear to be superior to currently used algorithms even if there are no faults.

The following are special awards received during the period:

- Lynch was chosen to deliver the keynote address at last summer's symposium on Principles of Distributed Computing.
- Meyer was chosen to present an invited lecture at the Third IEEE Symposium on Logic in Computer Science, July 1988.
- Sipser was chosen as the principal lecturer in the American Mathematical Society Conference on Circuit Complexity to be held this August in Chicago.

Baruch Awerbuch

Awerbuch has been working on designing efficient and reliable distributed protocols, with emphasis on issues related to dynamic networks.

He put a great deal of effort into development of an efficient compiler for dynamic network protocols. In [25] he used techniques of amortized analysis to improve the best known compiler for asynchronous protocols. Together with Sipser [32], he introduced a new concept of *dynamic synchronizer* which allows us to apply static synchronous protocols in a dynamic asynchronous network. This protocol is very fast, requiring $O(1)$ time overhead, thus showing that dynamic asynchronous networks are as fast as static synchronous ones. Finally, working with Afek and Moriel (Tel-Aviv) [3], he discovered a compiler whose overheads depend exclusively on the overheads of the origir ' protocol.

Awerbuch also worked on many specific problems in dynamic networks. Together with Shavit and Mansour [31], Awerbuch discovered the first polynomial solution to the end-to-end communication problem. This is one of the basic network problems; it was conjectured in [4] that it has no polynomial solution. Together with Goldberg (Stanford), Luby (ICSI) and Plotkin (Stanford) he found a new technique [28] for removing randomness from distributed computing that has yielded fast deterministic algorithms for Maximal Independent Set, $\Delta + 1$ Coloring and Breadth First Search. Together with Kutten (IBM Yorktown) and Cidon (IBM Yorktown), he discovered an efficient algorithm for maintaining a tree in a dynamic network. Together with Goldreich and Herzberg (Technion) [29], he developed a quantitative framework for analyzing performance of broadcast protocols in dynamic networks.

Another area of Awerbuch's research has been in distributed graph algorithms. Together with Bar-Noy (Stanford University), Linial (IBM Almaden Research Center), and Peleg (Stanford University) [27], he discovered new routing schemes that use only bounded space, have low communication overhead, can be constructed online, work for weighted graphs, and do not require changes in node identities. He discovered a new efficient BFS and Shortest Paths algorithm [26], which is efficient both in time and in communication. This algorithm has an interesting recursive structure. Together with Goldreich (The Technion), Peleg (Stanford University), and Vainish (The Technion) [30], Awerbuch studied performance of broadcast protocols in point-to-point networks.

Peter Elias

The paper on the zero-error capacity of a binary channel under jamming using list decoding, which was accepted for publication at the time of the last annual progress report, has since appeared [98]. Its appearance led to correspondence with Körner, who has been working on related topics with Marton and Simonyi. Their work arose from a paper on hashing by Fredman and Komlos [114]. They published one paper [185] and submitted two more, which include new results relevant to zero-error capacity under list decoding.

The second paper mentioned in the last progress report, which does not have to do with zero-error capacity but with error-correcting codes under list decoding, has appeared as a technical report and has been submitted for publication [99].

Current work explores iterative coding schemes. These schemes generate codes which differ from typical error-correcting block codes in that they are not guaranteed to correct *all* sets of less than k errors out of n for some integers k,n but only *most* such sets. Only codes with this property can be used to communicate at rates near channel capacity: as discussed in [98] and [99], the capacity of a channel subject to a jammer who can alter any k symbols out of n is significantly less than that of a channel in which bits are subject to statistically independent errors with probability k/n.

The first analysis of these codes appeared in [97]. It showed that they could be used to transmit without error at a positive rate, by using check symbols to correct each row of transmitted symbols, rows of check symbols to correct each column in a two dimensional array, layers of check symbols to check preceding layers in a rectangular solid, and so on. The fraction of the symbols used for checking is less than 1 in the limit if the sizes of successive dimensions increase, e.g., in a geometric series.

In [97] each order of check symbols is used only once and then discarded. That sufficed to show that communication at a positive rate is possible, but the proof gives a rate substantially below channel capacity. The rates of iterated codes come much closer to capacity when lower order check bits are used to make further corrections after each use of higher order check bits, and the process is continued until a stable state is reached. Since statistical independence disappears after such recycling, getting tight bounds on the amount of improvement is difficult. Both analysis and simulation are being used to explore this domain.

Shafi Goldwasser

Goldwasser's work focused on designing efficient digital signature schemes and on designing multi-party secure cryptographic protocols.

Beaver and Goldwasser [36] designed a protocol for n processors, a majority of which can be faulty, to compute any polynomial time function defined on the processors private inputs. The function is computed preserving privacy. Namely, no coalition of faulty processors can discover more about non-faulty processors inputs than implies by the function value. Moreover, the faulty processors can find out the function value "if and only if" the non-faulty processors find out the function value, in a strong probabilistic sense. This is the first solution in the case where the faults constitute more than a majority of the network processors.

Ben-Or, Kilian, Goldwasser, and Wigderson [43] designed two extremely efficient user identification methods (using no modular multiplications and based on the difficulty of the NP-complete subset-sum problem). These schemes work in the two prover interactive proof model introduced by the same authors in '88. Namely, the prover (e.g., Bank card holder) is split into two agents, and the verifier (e.g., the Bank teller machine) guarantees that the two agents can not transfer information to each other during the identification process.

Bellare and Goldwasser [37] introduced new paradigms for digital signatures and message authentications which are a complete departure from the digital signatures schemes based on Diffie-Hellman trapdoor function model or the recent digital signature scheme of Naor-Yung. The new scheme is based on the use of random functions and noninteractive zero-knowledge proofs.

Goldwasser has also been developing a monograph of lecture notes in cryptography, an outgrowth of her lectures in the MIT cryptography and cryptanalysis class. Goldwasser chaired the CRYPTO-88 conference held in Santa Barbara in August 1988. She was a member of the STOC 1989 conference committee, and together with Rivest, wrote a survey article on cryptography for the handbook on computer science.

Tom Leighton

Together, Leighton and his students made solid progress on packet routing algorithms, fault tolerance in networks, and on graph embedding problems. At this point they are getting close to asymptotically optimal results that also appear to work well in reality. In fact, the highlight of the coming summer and fall will be to help design and layout a multibutterfly network for Tom Knight's new machine. With a little luck, theory will be able to play an important role in the development of a state of the art machine. They are also working with Bill Dally and his students to see if theory can be helpful with the routing protocols on his new machine, and have been talking with Alan Baratz about the possibilities of implementing some of the new theory routing algorithms on the GF11 so that it can become a general purpose routing machine.

Another highlight of the coming year will be the new ACM Symposium on Parallel Algorithms and Architectures that Leighton has been helping to organize. The first meeting will be in Santa Fe in mid-June, and there should be a large contingent from MIT at the meeting. Papers to be presented range from theory to practice, and the meeting should provide a good forum for interaction between people who think about parallel machines, those who build them, and those who use them. The 1990 meeting is in Crete, so now would be a good time to start thinking about submitting a paper!

Maggs, Rao, Koch, and Newman are students getting their Ph.D.'s this year.

In addition, Leighton is continuing work on his book on parallel computation. He expects to have Volume I done by early next year.

Charles E. Leiserson

Leiserson returned in January from a leave of absence at Thinking Machines Corporation, where he worked on the design of a parallel computer. He was an invited speaker at the 25th Anniversary Symposium for Project MAC at MIT, and at the Decennial Caltech VLSI Conference. He served on the program committee for the IEEE Foundations of Computer Science Conference. He also served on the first program committee for the ACM Symposium on Parallel Algorithms and Architectures.

Leiserson has spent much of his time in the past year working on a textbook entitled *Introduction to Algorithms*, coauthored with Cormen and Rivest. The textbook attempts to provide a rigorous, but elementary, introduction to the area of analysis of algorithms. It will be published jointly by MIT Press and McGraw-Hill later this year.

Two of Leiserson's Ph.D. students completed their degrees in the past year. Plotkin's thesis is entitled *Graph-Theoretic Techniques for Parallel, Distributed, and Sequential Computation.* Plotkin assumed a postdoctoral position at Stanford and will be an assistant professor in the fall. Blelloch's thesis is entitled *Scan Primitives and Parallel Vector Models.* Blelloch accepted an assistant professorship at Carnegie-Mellon University.

Three students completed their Master's degrees under the supervision of Leiserson. Ishii's thesis, *A Digital Model for Level-Clocked Circuitry*; Park's thesis, *Notes on Searching in Multidimensional Monotone Arrays*; and Fried's thesis, *VLSI Processor Design for Communication Networks.*

Leiserson has also been supervising Cormen, Greenberg, Kipnis, Maggs, Phillips, and Papaefthymiou.

Nancy A. Lynch

Please see entry under the chapter on Theory of Distributed Systems.

Albert R. Meyer

Meyer's research has focused on semantics and logic of programming languages. During the past year, he worked on the following:

Research Topics

- *Semantics of Concurrency*: Meyer, with Bloom and Istrail (Wesleyan), question the foundations of Hoare's CSP and Milner's CCS theories of concurrency [51][53][52]. They propose a new notion of process equivalence and show it lies strictly between that of CSP and CCS. See the report of Bard Bloom for more complete discussion.
- *Semantics of Terminating Evaluation:* Research with Bloom, Riecke, and Cosmadakis (IBM Watson Research Center) on the general connection between operational and denotational semantics, focusing on repairing the mismatch between semantics in which expressions M and $\lambda x.M$ mean the same thing, even though evaluation of M diverges but evaluation of $\lambda x.M$ terminates immediately, cf. [233][85][54]. See the report of Riecke.
- *Dataflow Semantics*: See the report of Rudich.
- *Theory of Sequential Functions*: See the report of Jim.
- *Type-checking for Records with Inheritance*: See the report of Jategoankar.

Professional Activities

- Chairman, MIT Project MAC 25th Anniversary Celebration, October 1988.
- Conference Chairman, IEEE Symposium on Logic in Computer Science (LICS), Seattle, WA, May 1989.
- Moderator for three Computer Science research email forums on (1) types, (2) concurrency, and (3) logic.
- Member, Program Committee, International Symposium on Logic at Botik, Pereslavl-Zalessky, USSR, July 1989; "Kleene '90" Logic Symposium, Chaika, Bulgaria, June 1990.
- Thesis Supervision:

  PhD Bloom, expected September 1989.

  SM Riecke, completed January 1989 [273].
  Jategoankar, expected September 1989 [171].
  Jim, expected January 1990.
  Rudich, expected January 1990.

SB Ernst, completed May 1989 [100].
Lent, expected May 1990.
Siegel, expected September 1989.

- Editorial Activity:

  Editor-in-Chief, *Information and Computation*; Managing Editor, *Annals of Pure and Applied Logic*; Editorial Board Member, *SIAM Journal of Computing*, *Journal of Computer and System Sciences*, *Theoretical Computer Science*, and *Advances in Applied Mathematics*; Advisory Editor, *Handbook of Logic in Computer Science* and *Handbook of Theoretical Computer Science*; Co-editor, *Proceedings of Logic at Botik* [235]; MIT Press *Foundations of Computing Series* Co-editor; MIT Press Editorial Board Member.

Silvio Micali

Micali's work focused on cryptography and zero-knowledge proofs. In particular, the following results were obtained:

1. Goldreich, Micali, and Wigderson previously proved that all theorems in NP possess a zero-knowledge proof. Extending that work, [41] showed what can be efficiently verified can be proven in zero knowledge.
2. [237] constructed a very efficient "password" scheme. The person seeking identification is required to perform the equivalent of two multiplication modulo on an integer that is hard to factor. These special "passwords" are hard to compromise both by someone simply listening to the identification process and by the password verifier herself.

Ronald L. Rivest

Rivest's work focuses on the theoretical aspects of machine learning.

Rivest is continuing to work with Schapire on problems related to the inference of finite automata. Their motivation has been the "artificial intelligence" problem faced by a robot placed in an unfamiliar environment with no *a priori* knowledge of its world. The goal of the robot is to learn the structure of its environment through systematic experimentation.

Schapire and Rivest [274] have been developing an interesting extension to Angluin's finite automaton inference procedure [15]. The new algorithm can infer an automaton even when no "reset" is available (i.e., there is no means of bringing the automaton back to the start state), and can be used for inferring automata using either the global state-space representation or the diversity-based representation previously developed by Rivest and Schapire. The algorithm has been implemented and seems quite efficient in practice.

Together with Goldman and Schapire, Rivest studied the problem of "learning a binary relation" [130]. In this problem, the entries of a matrix representing a binary relation are repeatedly probed. Before each probe, the "learner" must predict the value of the matrix entry about to be probed. The goal of the learner is to make as few prediction errors as possible. In order to model the natural "structure" that may be present in many binary relations, such structure being what gives the learner the leverage needed to make fewer than the maximum possible number of prediction errors, it is assumed that there are only a small number k of different row types. Algorithms are developed and analyzed that make a small number of errors in this case, and some interesting lower bounds (based on the existence of projective geometries) are proved.

Goldman and Rivest [129] also worked on the problem of efficiently implementing the "halving algorithm." The halving algorithm applies to situations (like the relation-learning problem of the last paragraph) where the learner must predict the classification of each instance before being told the true classification, and where the learner's goal is to minimize the number of prediction errors made. The halving algorithm (due to Barzdin and Freivalds [35], and refined by Littlestone [215]) predicts in according to the majority of the hypotheses consistent with all previous data; when a prediction error is made it therefore reduces by half the number of consistent hypotheses remaining. Based on a proposal by Warmuth, Goldman and Rivest have investigated the use of approximate counting scheme in order to implement approximations to the halving algorithm. This idea can be made to work out, and can be applied to problems such as learning a total order. (This problem is then rather like the problem of sorting, where an adversary gets to pick which elements are to be compared next, and where you must predict the outcome before each comparison is made.)

Sloan has finished up his Ph.D. under Rivest's supervision [290]. His thesis explores a number of fascinating issues and topics in machine learning theory, such as the effect of noisy data on learnability, techniques for learning a complicated concept reliably and usefully by learning it "gate by gate" (subconcept by subconcept), and methods for combining classical Bayesian inference with computational complexity considerations.

Linial, Mansour, and Rivest extended and presented their work showing that a finite Vapnik-Chervonenkis dimension is not a limitation for learning a concept class, if the size of the data sample used for learning can be adjusted dynamically as learning proceeds [209]. Intuitively, an algorithm can dynamically request more data when it discovers that the concept being learned is "complex."

Blum finished up his Master's thesis [55] under Rivest's supervision, and the work he and Rivest have done on the complexity of training even very simple neural networks was presented at NIPS [57]. The basic result is that training a three-neuron neural network is NP-complete.

Under Rivest's supervision, Perugini has experimentally examined the effect of training set data size on the efficacy of the "back-propagation" training algorithm for neural nets [259]. The results were not crisp, but some interesting pathologies were uncovered.

Together with Cormen and Leiserson, Rivest worked on a introductory text on algorithms [84]. This text should be suitable for both introductory undergraduate and introductory graduate students; it should be out later this year.

David B. Shmoys

Shmoys studied a wide range of questions in the design and analysis of efficient algorithms. He continued his work in the design and analysis of approximation algorithms, as well as in the design of parallel algorithms for graph problems.

One of the most important algorithms used in the solution of the traveling salesman problem is a procedure due to Held and Karp [157] that produces an extremely tight lower bound on the value of the optimal solution. With Williamson [287], Shmoys considered this procedure as an approximation algorithm for the value of the optimal TSP solution. First, they showed that the algorithm has an important monotonicity property, in the sense that the bound delivered for a subset of the input is no more than for the entire input. This property makes it possible to prove that the procedure delivers a value at least 2/3 of the optimal value. Unlike Christofides' algorithm, which is the best known approximation algorithm for the problem (and guarantees identical performance), this bound is not known to be tight.

One major area of Shmoys' research is in the area of the theory of scheduling. Together with Lawler (UC/Berkeley), Lenstra (CWI) and Rinnooy Kan (Erasmus) [197], he wrote a survey article of the field. This survey was written as part of the preparation for a book on this subject by these authors.

With Hall (Sloan School/MIT) [143][142], Shmoys has been considering a variety of approximation algorithms for scheduling problems. In particular, he has been studying the effect of precedence constraints and related timing constraints on the possibility of obtaining good approximate solutions. Hall and Shmoys [143] consider the problem of scheduling n jobs on a single machine, where each job j has a specified release date r_j before which is cannot be processed, a time p_j that specifies the amount of (continuous) processing required, and a deadline d_j. (For technical reasons, the deadlines are non-positive.) If the lateness of a job is the difference between the time that a job completes processing and its deadline, the aim is to find a schedule that minimizes the total lateness. For the variant of the problem without precedence constraints, a polynomial approximation scheme is obtained. For the problem with precedence constraints, they give an algorithm that delivers a solution that finishes within a factor of 4/3 the optimal time (improving on the previous best algorithm that only came within a factor of 2). This represents an interesting breakthrough of a "factor of 2" barrier that is prevalent in approximation algorithms for precedence constrained scheduling problems. Also with Hall [142], Shmoys considers the natural generalization of the previous work to the case when there are parallel identical machines to do the processing. For this problem without precedence constraints, a polynomial approximation scheme was obtained. With precedence constraints, an algorithm that delivers a solution at most a factor of 2 more the optimal was obtained.

In the area of parallel graph algorithms, together with Goldberg (Stanford), Plotkin (Stanford), and Tardos [124], Shmoys considers the question of parallel algorithms for bipartite matching. By using techniques developed for general purpose sequential algorithms for linear programming, so-called interior-point methods, they obtain an algorithm that requires only $O^*(\sqrt{m})$ steps on a polynomial number of processors, where m denotes the number of edges in the graph, and O^* indicates that lower order polylogarithmic factors have been ignored.

Michael Sipser

Sipser is continuing his work on lower bounds in complexity theory and the structure of complexity classes.

One of the important achievements of the past year was the construction of an oracle collapsing the probabilistic time hierarchy done jointly with Fortnow [112]. Time hierarchies for deterministic and nondeterministic computation are among the earliest results proved in complexity theory. They show that if one is allowed a little more time then one can solve a larger class of problems. Oddly, this has never been established for probabilistic computation. It is possible that any problem solvable in probabilistic polynomial time can also be solved in probabilistic linear time, surprising though this would be. Our result shows why this problem has remained open. The existence of our oracle indicates that the techniques of recursive function theory, which solved the previous cases, are insufficient to solve this case. Fortnow is receiving his Ph.D. this year under Sipser's guidance.

Sipser also considered some problems in the theory of distributed computing. Together with Awerbuch, he gave a method which facilitates the design of network protocols [32]. Using this method, one can first design a protocol to run on a static, synchronous network and then automatically convert it to run on a dynamic, asynchronous network. The former network model is a simpler one on which to conceive designs, whereas the latter model is more realistic.

Together with Boppana, Sipser prepared a definitive survey on lower bounds on the circuit complexity of boolean functions [60]. This will appear in the forthcoming *Handbook of Theoretical Computer Science.* Sipser was selected to be the principal speaker at an American Mathematical Society conference on circuit complexity. He will prepare a monograph of these lectures to be included in the *AMS CBMS* series.

Éva Tardos

Tardos has been working on combinatorial optimization problems. Together with Goldberg and Plotkin from Stanford and Shmoys from MIT [124], she developed an $O^*(\sqrt{m})$ time algorithm, where n and m denotes the number of nodes and edges of the input graph and an algorithm is said to run in $O^*(f(n))$ time if it runs in $O(f(n)\log^k(n))$ time for some constant k. In this paper, interior-point methods for linear programming, developed in the context of sequential computation, are used to obtain a parallel algorithm for the bipartite matching problem. The results extend to the weighted bipartite matching problem and to the zero-one minimum-cost flow problem, yielding $O^*(\sqrt{m}\log C)$ algorithms, where it is assumed that the weights are integers in the range $[-C \dots C]$ and $C > 1$. These results improve previous bounds on these problems and introduce interior-point methods to the context of parallel algorithm design.

In a joint paper with Plotkin from Stanford [266], Tardos gave an improved dual network simplex algorithm. A simplified version of Orlin's [248] strongly polynomial minimum-cost flow algorithm is developed, and it is shown how to convert it to a dual network simplex.

The pivoting strategy leads to an $O(m^2 \log n)$ bound on the number of pivots, which is better by a factor of m compared to the previously best pivoting strategy due to Orlin [247]. Here n and m denotes the number of nodes and arcs in the input network.

In a joint paper with Frank from Budapest and Nishizeki, Saito, and Suzuki from Tokyo, Tardos developed simple efficient algorithms for the routing problems around a rectangle. These algorithms find a routing with two or three layers for two-terminal nets specified on the sides of a rectangle. The minimum area routing problem is also solved. All algorithms run in linear time. The minimum area routing problem was previously considered by LaPaugh and Gonzalez and Lee. The algorithms they developed run time $O(n^3)$ and $O(n)$, respectively. The simple linear time algorithm is based on a theorem of Okamura and Seymour, and on a data structure developed by Suzuki, Ishiguro, and Nishizeki.

Tardos has also written two surveys this year. A general survey on complexity theory for *The Handbook of Combinatorics* [286] jointly with Shmoys from MIT, and a survey on the recent development in the theory of network flows [125] jointly with Goldberg from Stanford and Tarjan from Princeton.

13.2 Student, Research Associate, and Visitor Reports

Javed A. Aslam

Aslam has been working with Rivest on algorithms for machine learning. Specifically, he has been studying the radial mapping problem where a device must infer the shape of its surroundings by rotating in place and taking distance measurements. Relevant cases studied have included those where angular positioning error and distance measurement error are present in varying degrees. Aslam recently began work on the inference of Markov chains, and plans to continue this work with Rivest over the summer.

Mihir Bellare

Basic cryptographic primitives such as zero-knowledge proofs and oblivious transfer have classically relied on *interaction* between the parties involved. A part of Bellare's work has focused on a new *public key* model in which such interaction can be removed.

Bellare and Micali [38] proposed a method via which a collection of users may first establish public keys and then be able to accomplish oblivious transfer *without* interaction. Using earlier work of [236], this yields noninteractive methods for zero-knowledge proofs.

Bellare and Goldwasser [37] demonstrated the wide applicability of such noninteractive zero-knowledge proofs by using them to get simple and efficient schemes for digital signatures and message authentication. A feature of this work was an implementation of noninteractive zero-knowledge proofs which could be checked by any user in the system rather than by a single recipient.

In further work related to the role of interaction in zero knowledge proofs, Bellare, Micali and Ostrovsky [39] showed that the languages of graph isomorphism and quadratic residuosity have *constant round* perfect zero-knowledge interactive proofs. They also provided a general mechanism to collapse rounds in a statistical zero-knowledge proof while preserving the statistical zero knowledge, given some standard cryptographic assumption.

Bonnie Berger

Berger has been working on removing randomness from parallel and sequential algorithms. This involves coming up with a randomized algorithm for a problem, if one does not exist, and devising or using known techniques to remove this randomness.

Berger began this work at Bell Labs last summer when, with Shor, she devised a randomized sequential algorithm for the acyclic subgraph problem (the dual of the feedback arc set problem) and used known, highly sequential techniques to convert it to a deterministic one, thereby achieving tight bounds deterministically for the problem [48]. This work also included an RNC algorithm for the problem which, by applying techniques explored in her subsequent work, Berger is attempting to convert to a deterministic one.

Berger's subsequent work centered around removing randomness from parallel algorithms.

Berger and Rompel [46][44] developed a general framework for removing randomness from randomized NC algorithms whose analysis uses only polylogarithmic independence. Previously, no techniques were known to determinize those RNC algorithms depending on more than constant independence. One application of their techniques is an NC algorithm for the set discrepancy problem, which can be used to obtain many other NC algorithms, including a better NC edge coloring algorithm. As another application of their techniques, they provided an NC algorithm for the hypergraph coloring problem. This work has been chosen for the FOCS '89 Machtey Award.

Berger, Rompel, and Shor [47] gave NC approximation algorithms for the unweighted and weighted set cover problems. Their algorithms use a linear number of processors and give a cover that has at most $\log n$ times the optimal size/weight, thus matching the performance of the best sequential algorithms. Previously, there were no known parallel algorithms for the general set cover problem. Berger, Rompel, and Shor devised a randomized algorithm, depending on only pairwise independence, and then converted it to a deterministic one. The difficult part here was coming up with the randomized algorithm. Furthermore, they applied their set cover algorithm to learning theory, giving an NC algorithm to learn the concept class obtained by taking the closure under finite union or finite intersection of any concept class of finite VC dimension which has an NC hypothesis finder. In addition, they gave a linear processor NC algorithm for a variant of the set cover problem first proposed by Chazelle and Friedman, and used it to obtain NC algorithms for several problems in computational geometry.

Bard Bloom

Bloom, working with Meyer and Istrail (Wesleyan) is studying the denotational semantics of parallel and nondeterministic processes. Dana Scott's very successful models for the semantics of sequential, deterministic programs do not extend naturally to the more general domain. There are a number of proposals for a replacement; Meyer and Bloom are investigating several of these models. One central question in semantics is, "when shall we consider two programs equivalent?" Two proposed notions are *trace congruence* (used in Hoare's language CSP and variants) and *bisimulation* (used in Milner's SCCS). Bloom, Meyer, and Istrail have found an extension of SCCS in which the two notions coincide. The new operation is somewhat peculiar in nature; they have shown that no finite set of operators defined in a clean way can cause the two to coincide. Similarly, bisimulation cannot be understood as equivalence with respect to any set of reasonable experiments. It can be understood in a probabilistic setting; however, the translation from the usual setting to the probabilistic one is not effective.

This work lead to a notion of "ready simulation" which seems to have the same sorts of formal properties as bisimulation (various alternate definitions, complete axiomatizations and polynomial time decision procedures for finite processes, and so forth), but can also be understood as congruence with respect to a fairly reasonable language.

A classic paper in denotational semantics (Gordon Plotkin's *LCF Considered as a Programming Language*) gives two kinds of semantics for a simple but extremely powerful language based on typed lambda calculus. One semantics is *operational*, describing how a particular interpreter computes; the other kind is *denotational*, assigning meaning to the programs in moderately familiar mathematical terms, using several varieties of Scott domains. The paper shows that the two semantics coincide in a weak sense (*computational adequacy*; two integer terms evaluate to the same constant if they have the same denotational meaning), but not in a stronger sense (*full abstraction*: two routines behave identically in all contexts if they have the same denotational meaning). The programming language can be extended by the addition of a "parallel conditional" such that the extended language is fully abstract for one of the denotational models. The classic paper shows that this extension is not fully abstract for the other languages.

However, one of the other denotational models (Scott domains built from complete lattices rather than cpo's) is mathematically appealing, and it is somewhat surprising that the classic paper did not find a fully abstract extension of LCF using this model. However, this is not the author's oversight. Bloom has shown that there is *no* fully abstract extension of LCF with a reasonable evaluator for which this model is fully abstract, where "reasonable" means that an arithmetic expression can evaluate to at most one value. If the evaluator is not required to be reasonable in this sense, there is a simple extension of LCF after the spirit of the classic paper which is fully abstract for the lattice model. If the evaluator is allowed to have a technically peculiar property, it can be made fully abstract for virtually any model of the typed lambda calculus.

Bloom and Riecke have been investigating similar questions for the so-called "lifted Scott domains." Ordinary functional languages exhibit some behavior on higher-order terms: if a program evaluates to a function, it stops and prints "function"—even if the function will always diverge when applied to any argument. In ordinary Scott domains, there is no semantic difference between the function which always diverges given any argument, and a divergent computation of functional type. Lifted domains repair this deficiency. Bloom and Riecke have achieved a close correspondence between operational and denotational semantics for this setting, and are investigating axiom systems.

Avrim Blum

Blum has been working in two main areas this past year and has also finished his Master's thesis [55] under Rivest's supervision.

He continued his work with Rivest on problems in computational learning theory—in particular, computational complexity issues in the training of neural networks. One result of this work is a proof that training a very simple neural network with only three computational nodes is NP-complete. This work was presented at the NIPS and COLT conferences [57].

Blum has also been working on approximate graph coloring. The 3-coloring problem is one of the most well known NP-complete problems, but there is an enormous gap between the results achieved by the best approximation algorithms for this problem and the best lower bounds known. Blum devised a new approximation algorithm [56] that reduced this gap somewhat and introduced different techniques for attacking this problem.

Thomas H. Cormen

Cormen continued his work on the textbook *Introduction to Algorithms* with Leiserson and Rivest. He plans to start working on parallel computing research over the summer.

Lenore Cowen

Cowen continues work with Goldwasser on two areas: key exchange protocols and information theoretic properties of private functions.

Claude Crépeau

Crepeau's current research interest is mainly the study of two-party cryptographic protocols. His earlier study of disclosure protocols [63] [64] evolved in a series of results [86][89][88][177][87] essentially stating that very complex two-party protocols known as *fair oblivious circuit evaluation* (see [87] for definition) can be achieved from very simple devices. Such a device can be a simple noisy channel, for instance. Another such possible device follows the lines of Bennett and Brassard and rely on the correctness of quantum physics. This work was accomplished in part while Crepeau was visiting Aarhus University (Denmark) in the summer.

Crepeau is currently completing his Ph.D. thesis, that will cover some recent material selected from the above papers. He is expected to defend his thesis during the summer.

Zero-knowledge protocols is another of Crepeau's favorite research topics. While visiting IBM Almaden Research Center last summer, he contributed two papers on this subject [65][62]. These two papers are followup to [61], in the fact that they are concerned with a model where the prover involved in the protocol is computationally bounded.

Aditi Dhagat

During fall 1988, Dhagat was a teaching assistant for the graduate course in theory of computation taught by Sipser. During the year, she worked with Sipser in complexity theory and cryptography, trying to construct a pseudorandom number generator secure against monotone circuits without any unproven assumptions. In the process, they looked at monotone statistical tests and showed that there exist exponential size monotone statistical tests which break the security of the Nisan-Wigderson generator based on parity. They have also shown that if there exist monotone functions which are hard to approximate for polynomial size monotone circuits, then there exist pseudorandom number generators secure against polynomial size monotone circuits.

Dhagat plans to continue to work on this question during the summer of 1989.

Michael Ernst

Ernst became a graduate student at MIT in January 1989. He worked under Meyer's supervision to prove a monotone model adequate for recursive program schemes. In order to prove adequacy, most proofs in the literature directly use a stronger continuous model which simplifies the proof and which implies the weaker result; the typical approach is via Tait's method of computability [265][296]. The introduction of continuity is poorly motivated from an expository and pedagogical viewpoint; we would hope to be able to show the result directly [233].

Ernst and Meyer [100] found that this was not possible; although they were able to produce a clear exposition of the concept, at one crucial point continuity was required. While the result holds for the monotone model without mention of continuity, a weaker assumption of monotonicity in the proof leads to a failure of the result.

Ernst spent much of 1989 finishing up his undergraduate requirements; he plans to get started on his SM thesis during the upcoming year.

Lance J. Fortnow

Working with Sipser, Fortnow examined the relationship between probabilistic polynomial time and probabilistic linear time. They showed [112] the existence of an oracle under which the two classes are identical. This result means the techniques of separating the deterministic and nondeterministic time hierarchies will not work for probabilistic computation. They also show many other results relating to probabilistic computation and linear time.

During the spring of 1989, Fortnow spent the semester writing his thesis [111] and looking for a job.

Jeff Fried

Fried continued research on the architecture, design, and analysis of communication networks for use in parallel computers and telecommunications. He completed a Master's thesis [117], supervised by Leiserson, which includes two switch designs for such networks [120][116]. Followup work in this area has included an improved circuit design for one of the VLSI functions used in these designs [118], and a study of some of the modularity tradeoffs found in sparse circuit-switched interconnection networks [119].

Fried is currently working on a number of problems related to the architecture and control algorithms needed for high performance communication networks. This work includes a study of the impact of synchrony on the performance of distributed algorithms, and design studies of a VLSI packet router for use in broadband networks [115].

Sally A. Goldman

Goldman has been working with Rivest on studying learning algorithms for concepts that have polynomial sized instance spaces [129][130]. They have focused on polynomial prediction algorithms in which the learner predicts a value for each entry in the instance space and then receives feedback as to whether the prediction was correct. They consider the worst case mistake bounds under several models for the selection of the instances. Often, good mistake bounds are obtained by the halving algorithm. They discuss an approximate halving algorithm and show how a fully polynomial randomized approximation schemes can be used to implement (with high probability) the approximate halving algorithm. They demonstrate these techniques on the problem of learning a total order on a set of n elements.

Goldman has also been working with Rivest and Schapire on the particular problem of learning a binary relation between n objects of one kind and m of another [130]. This can be viewed as the problem of learning an $n \times m$ binary matrix. Here, the instance space contains the elements of the matrix and is thus of polynomial size. They present numerous upper and lower bounds on the number of mistakes that prediction algorithms can make under different models for the selection of the instances.

Goldman has also done some research in the field of computational geometry. In particular, she developed an algorithm to compute the greedy triangulation of an arbitrary point set that takes $O(n^2 \lg n)$ time and $O(n)$ space [128]. In January, Goldman participated in the robot building project lead by Schapire.

Ronald I. Greenberg

Greenberg worked on three main topics during the past year: networks for general purpose parallel computation, multi-layer channel routing, and bounds on the area for VLSI implementations of finite-state machines.

Recent work on networks for general purpose parallel computation is reported in [136]. This paper provides several extensions and generalizations of earlier work on the problem of designing "universal" networks which can simulate any other network of comparable physical size with only polylogarithmic overhead in simulation time.

On the topic of multi-layer channel routing, Greenberg has been seeking improvements upon algorithms recently developed with Ishii and Sangiovanni-Vincentelli (University of California, Berkeley) for the program MULCH [137]. The basic approach of MULCH is to divide a multi-layer problem into essentially independent subproblems of one, two, or three layers. A main step in MULCH is to greedily partition the nets once a set of layer groups has been determined. As each net is considered, it is assigned to the group where the resulting subproblem seems to be the one requiring the least channel width. For testing the required channel width of single-layer partitions, Greenberg and Miller Maley (Princeton University) have devised algorithms which are more efficient than naive approaches involving complete routing of the layer. Greenberg is also developing "incremental" algorithms to quickly determine the effect on certain subproblem characteristics when a new net is added, by taking advantage of knowledge derived from earlier computations on the subproblem.

Finally, Greenberg and Mike Foster (Columbia U. and NSF) have derived lower bounds on the area required for VLSI layout of finite-state machines [113]. These lower bounds show that naive layout approaches are optimal in the worst case.

Michelangelo Grigni

Grigni is a third year graduate student supervised by Sipser. His thesis research considers the construction of fast robust broadcasting networks, continuing work begun with Peleg [139] of the Weizmann Institute. Current work with Bertsimas of the Sloan School extends their recent result [50] on the suboptimality of the space-filling curve heuristic for the Euclidean TSP problem. Other work with Bertsimas includes a survey of various #NP-complete problems. Grigni continues searching for new attacks on the matrix multiplication exponent problem.

Carolyn M. Haibt

Haibt spent most of the year on coursework, but also continued work with Tardos. They are currently working on algorithms for the generalized network flow problem. This is a generalization the maximum flow problem, where each edge has an associated gain factor, and flow is multiplied by this factor when it passes through an edge.

Mark D. Hansen

Hansen has been studying graph embeddings with applications to parallel processing problems. In [153], he examines the problem of finding optimal geometric embeddings in the plane and higher dimensional spaces. Given an undirected graph G with n vertices, and a set P of n points in R^d, the *geometric embedding problem* consists of finding a bijection from the vertices of G to the points in the plane which minimizes the sum total of edge lengths of the embedded graph. In general, this problem is NP-complete as it contains the Euclidean traveling salesman problem as a special case. Hansen gives approximation algorithms for embedding many of the important graphs studied in the theory of parallel computation. He presents fast algorithms for embedding d-dimensional grids in the plane which are within a factor of $O(\log n)$ times optimal cost for $d > 2$ and $O(\log^2 n)$ for $d = 2$. He also shows that any embedding of a hypercube, butterfly, or shuffle exchange graph must be within an $O(\log n)$ factor of optimal cost. When the points of P are randomly distributed or arranged in a grid, he is able to use the results of Leighton and Rao [202] to give a polynomial time algorithm which can embed arbitrary weighted graphs in these points with cost within an $O(\log^2 n)$ factor of optimal.

Hansen shows how the algorithms developed in [153] for geometric embeddings can be used to give solutions which are within an $O(\log^2 N)$ factor of optimal to problems of performance optimization for array-based parallel processors in the following areas: communication load balancing, dynamic allocation of jobs to processors, reconfiguring around faults, and simulating other architectures. He also indicates some applications to wafer scale integration problems and the dynamic configuration of distributed computing networks.

Working with Leighton, Hansen was able to apply some of the techniques developed in [153] to give an $N^{O(\sqrt{N})}$ time algorithm for solving the Euclidean traveling salesman problem. The previous best running time for this algorithm was $O(\log N 2^N)$. A year earlier Smith [293] independently gave an algorithm with the same running time, using different techniques involving the Lipton-Tarjan planar separator theorem [211]. Hansen and Leighton are currently investigating the possibility of developing practical heuristics for solving Euclidean TSP using the ideas in these two algorithms.

Alexander T. Ishii

Ishii completed his Master's thesis [168], which describes his models for VLSI timing analysis. The model maps continuous data domains, such as voltage, into discrete, or *digital*, data domains, while retaining a continuous notion of time. The majority of the thesis concentrates on developing lemmas and theorems that can serve as a set of "axioms" when analyzing algorithms based on the model. Key axioms include the fact that circuits in our model generate only well defined digital signals, and the fact that components in our model support and accurately handle the "undefined" values that electrical signals must take on when they make a transition between valid logic levels. In order to facilitate proofs for circuit properties, the class of *computational predicates* is defined. A circuit property can be proved by simply casting the property as a computational predicate.

Ishii has also been working with Greenberg and Sangiovanni-Vincentelli of Berkeley on a multi-layer channel router for VLSI circuits, called MULCH [137]. While based on the CHAMELEON system developed at Berkeley, MULCH incorporates the additional feature that nets may be routed entirely on a single interconnect layer (CHAMELEON requires the vertical and horizontal sections of a net be routed on different interconnect layers). When used on sample problems, MULCH shows significant improvements over CHAMELEON in area, total wire length, and via count.

Ishii continued work, begun with Maggs, on a new VLSI design for a high speed multiport register file. Design goals include short cycle time and single-cycle register window context changes. This research began as an advanced VLSI class project, under the supervision of Knight of the MIT Artificial Intelligence Laboratory.

Lalita A. Jategaonkar

Jategaonkar has been working jointly with Meyer on further developing research begun last year at Bell Laboratories with Mitchell. In [170], Jategaonkar and Mitchell develop an extension of the programming language ML in which a restricted object-oriented style can be achieved. In keeping with the framework of ML, a type derivation system and a type inference algorithm is presented. It is proved that the algorithm is sound and complete with respect to the type derivation system, and that it infers a most general typing of every typeable expression in the language. This research will comprise Jategaonkar's forthcoming Master's thesis.

In order to show that the type derivation system is "reasonable" in a precise, technical sense, Jategaonkar and Meyer have been developing an interpreter for this language. They aim to show that the interpreter satisfies certain desirable properties, and that the interpreter and the type derivation are well matched in the sense that no typeable expression in the language reduces to a type error. Jategaonkar is also interested in further extending ML to support subtyping of abstract types and recursive types. Another direction of research she is interested in pursuing is to develop a semantics for these extensions of ML.

Trevor Jim

Jim entered the department in September 1988. His previous work with Appel [16] on a novel code generator for the language ML was presented at POPL '89 in January.

Under the direction of Meyer, he has been studying the work of Berry and Curien [90][49] on models of PCF [265] based on stable functions and sequential algorithms. These models were developed as alternatives to the standard model, which contains troublesome "non-sequential" elements. Jim is trying to find extensions of PCF for which the alternate models are fully abstract.

Joe Kilian

Kilian spent most of his time working on his thesis, "Randomness in Algorithms and Protocols" [176], which he recently completed. He also did some work in efficient zero-knowledge interactive proofs, bounded interaction zero-knowledge proofs, noninteractive zero-knowledge proofs, multi-prover zero-knowledge proofs, space-bounded secure protocols, communication lower bounds for secret sharing, and IP v.s. AM.

A troubling issue in theoretical cryptography is the chasm between what is efficient in theory and what is efficient in practice. One area in which this gap is particularly large is in zero-knowledge proofs for NP predicates. Suppose one wishes to prove in zero-knowledge that some circuit, $C(x_1, \ldots, x_n)$, is satisfiable. The previously most efficient solutions to this problem ([61][167]) required the prover and the verifier to send $O(k|C|)$ bits back and forth per iteration of the protocol. Here, $|C|$ denotes the number of gates in the circuit C, and k denotes the security parameter. Using pseudorandom generators, Kilian [177] has exhibited a protocol in which the prover and the verifier communicate only $O(|C| + k^2)$ bits per iteration of the protocol. In real life circumstances, $|C|$ is likely to be very large, in which case this protocol should behave better in practice as well as in theory.

Zero-knowledge proofs typically require a great deal of interaction between the prover and the verifier. It is of both theoretical and practical interest to see how much interaction is truly needed, which led to the notions of *bounded interactive protocols* and *noninteractive protocols with a common random string.* In bounded interaction protocols, the prover and the verifier interact for time polynomial in the security parameter. After the interaction phase, the prover proves theorems to the verifier by sending him a letter in the mail. In a noninteractive protocol with a common random string, the prover and verifier do not interact at all, but are both presented with a uniformly distributed string of length polynomial in the security parameter.

Prior to Kilian's work, there existed three proposed protocols for these scenarios, due to Blum-Feldman-Micali [58], De Santis-Persiano-Micali [93], and Micali-Ostrovsky [250].

Kilian developed a very simple and efficient protocol for bounded interaction zero-knowledge proofs, and a provably secure protocol for noninteractive zero-knowledge with a common random string. Both of these protocols' security is based on reasonable cryptographic assumptions. His protocol for bounded interaction zero-knowledge proofs is more communication efficient than the best previously known interactive zero-knowledge protocols. In both of these protocols, the prover can prove polynomially many polynomial-sized theorems.

In [42], Kilian along with Ben-Or, Goldwasser, and Wigderson, developed a multiprover generalization of interactive proof systems. They showed that, informally, anything two provers could prove, they could prove in statistical zero-knowledge. Recently, Kilian has strengthened this result, showing that anything two provers could prove, they could prove in perfect zero-knowledge.

With Nisan, Kilian applied knowledge complexity notions from cryptography to space-bounded automata [179]. They developed protocols in this scenario for a number of cryptographic protocols: secret key exchange, bit-commital, secure circuit evaluation, and zero-knowledge proofs. In the space-bounded scenario, the security of these protocols may be proven without any assumptions whatsoever. Furthermore, these protocols are robust against adversaries who have asymptotically more space than used by the good players.

Nisan and Kilian also investigated upper and lower bounds for secret sharing. They consider schemes in which a bit b is shared among n players, such that,

1. A majority of the n players can reconstruct b; and

2. A nonmajority of the players cannot reconstruct any information about b.

They show a lower bound of $\Omega(n \log n)$ on the total number of bits that must be distributed amongst the n players. They also consider a weakened form of secret sharing, in which $2n/3$ players can reconstruct b, and $n/3$ players learn nothing. They use coding theory to prove the existence of secret sharing schemes that are more efficient than the lower bounds proven for the more stringent conditions.

A classic theorem of Goldwasser and Sipser [135] states that IP=AM. In other words, public coins are as powerful as private coins for interactive proof systems. Kilian found a very simple proof of this fact, using random selection techniques from [133]. This proof will be included in a paper on random selection, with Oded Goldreich, Johan Håstad, and Yishay Mansour.

Shlomo Kipnis

Kipnis has been investigating parallel architectures and interconnection networks. He is trying to further explore the power of bussed interconnection schemes in routing permutations and realizing various communication patterns. Bussed interconnection schemes and their relation to difference covers was explored by Kilian, Kipnis, and Leiserson in [178]. In addition, he is investigating various arbitration schemes for bussed based architectures.

Recently, he studied the problem of range queries in computational geometry. Range queries is a fundamental problem in computational geometry with applications to computer graphics and database retrieval systems. He compiled a survey report on three different methods for range queries in computational geometry [180].

Richard R. Koch

Koch's Ph.D. thesis [183] is a probabilistic analysis of routing on a parallel architecture. Koch analyzes the bandwidth of the butterfly network. In a dilated butterfly network, nodes are connected by parallel edges instead of just one edge as in the usual butterfly network. He proves a previous conjecture that the expected bandwidth of an N node dilated butterfly network is $\Theta(N(\log N)^{-\frac{1}{q}})$, where q is the number of parallel edges. He explores some implications of his results for design tradeoffs. He also develops interesting techniques for finding asymptotics for nonlinear systems of recurrences and many of the results appeared in [182].

In [184] Koch, Leighton, Maggs, Rao, and Rosenberg study the problem of emulating T_G steps of an N_G-node guest network on an N_H-node host network. Although many isolated emulation results have been proved for specific networks in the past, and measures such as dilation and congestion were known to be important, the field has lacked a model within which general results and meaningful lower bounds can be proved. They attempt to provide such a model, along with corresponding general techniques and specific results in this paper.

Dina Kravets

Kravets spent most of the year working with Aggarwal and Park on problems in computational geometry. In January, she finished her Master's thesis [191] which included the following results:

1. An algorithm to find all the farthest neighbors of every vertex on a convex n-gon in $\Theta(n)$ time.
2. An $O(n^2)$ algorithm to sort the distances of all the vertices of a convex n-gon with respect to each vertex of the convex n-gon.
3. An $O(kn \log k)$ time algorithm to find k *farthest vertices* for every vertex of a convex n-gon.
4. A worst-case optimal algorithm to sort a set of numbers given lower bounds on the ranks.

The first of these algorithms appeared in the *Information Processing Letters* [12]. Park and Kravets are planning to improve the third result and submit it to the ACM-SIAM Symposium on Discrete Algorithms.

Kravets is also looking at some problems in parallel computation and VLSI with Leighton.

Leonid A. Levin

The topic of Levin's research in 1988-89 may be called "Randomness in Computing." In [206], Levin and Venkatesan propose the first intractability results for random instances of NP problems. NP-complete problems should be hard on *some* (maybe extremely rare) instances. Generic instances of many such problems proved to be easy. This paper shows the intractability of *random* instances of a graph coloring problem. Applications of average case intractability are considered in two other papers: [132][166].

Blum and Micali [59] discovered permutations f with "hard-core" predicates $b(x)$ that cannot be efficiently guessed from $f(x)$ with a noticeable correlation. Both b, f are easy to compute. Yao [314] modifies any one-way permutation f into f^* which has a hard-core predicate. Its security may be lower than any constant power of the security of f and is too small for practical applications. Goldreich and Levin [132] prove that most linear predicates are hard-cores for every one-way function and have almost the same security. The result extends to multiple (up to the logarithm of security) hidden bits and has wide applicability to pseudorandomness, cryptography, etc.

Let an easily computable function f be one-way, i.e., for most x one cannot recover from $f(x)$ either (1) x by a polynomial time algorithm, or (2) an $x' \in f^{-1}(f(x))$ by a polynomial size circuit. In case (1), to exclude useless $f(x) = 0$, the difference between Shannon entropies of inputs and outputs of f is restricted to $O(1)$. Impagliazzo, Levin, and Luby [166] show, based on [132], that the existence of one-way functions in the sense (1) and (2) is necessary and sufficient for the existence of pseudo-random generators secure against feasible algorithms or circuits, respectively.

In [205], Levin compares probability distributions of computational objects. The usual distributions are concentrated on strings that differ little in any fundamental characteristic, except their informational size (Kolmogorov complexity). This property distinguishes a class of *homogeneous* probability measures suggesting various applications. In particular, it explains why the average case NP-completeness results are so measure independent, and offers their generalization to this wider and more invariant class of measures. It also demonstrates a sharp difference between pseudo-random strings and the objects known before.

Bruce Maggs

Maggs is studying the ability of a *host* network to emulate a possibly larger *guest* network [184]. His collaborators in this research are Koch, Tom Leighton, Rao, and Rosenberg. An emulation is *work-preserving* if the work (processor-time product) performed by the host is at most a constant factor larger than the work performed by the guest. Such an emulation is efficient because it achieves optimal speedup over a sequential emulation of the guest. Many work-preserving emulations for particular networks have been discovered. For example, the N-node butterfly can emulate an $N \log N$ node shuffle-exchange graph and vice versa. On the other hand, a work-preserving emulation may not be possible unless the guest graph is much larger than the host. For example, a linear array cannot perform a work-preserving emulation of a butterfly unless the butterfly is exponentially larger than array. These positive and negative results provide a basis for comparing the relative power of different networks.

Maggs is also studying algorithms for routing packets on faulty bounded-degree networks. With Leighton, he developed a scheme for routing N packets on an N-node multibutterfly network [303] in $O(\log N)$ steps even in the presence of many faulty nodes.

Yishay Mansour

Yishay Mansour has continued studying data transmission in communication networks. In a work with Schieber [229], they show lower bounds for communication over non-FIFO links. In a work with Herzberg and Goldreich [131] they give a randomized protocol for communication over non-FIFO links. In a work with Awerbuch and Shavit [31] they show how to achieve polynomial end-to-end communication.

In work with Linial and Nisan [208], they investigate constant depth circuit using the Fourier Transform. They are able to show a quasi-polynomial time algorithm for learning this class. Another work that is connected to learning is [40].

In a work with Schieber and Tiwari [231], they continue to develop techniques to prove lower bound for integer computations. The work with Schieber and Tiwari [230] tries to explore the complexity of approximating algebraic functions. In this work, techniques taken from Approximation Theory are used to derive lower and upper bound.

Mark J. Newman

Newman continued work on fault-tolerant strategies for parallel computation. With Hastad and Leighton [156], he demonstrated algorithms for reconfiguring hypercubes with faulty components. After reconfiguration, the hypercubes retain all computational power (within constant factors). The algorithms are successful with high probability, given that nodes and edges fail independently and with constant probability. They also showed how to route permutations on hypercubes even if a constant fraction of the cube's components have failed.

With Leighton, Ranade, and Schwabe [201], Newman also showed how a dynamically changing binary tree can be embedded in a hypercube so that computational and communication overhead are low. Specifically, they produced randomized algorithms which embed any growing and shrinking binary tree so that the resulting simulation requires only constant factor overhead, with high probability.

Noam Nisan

Nisan arrived as a postdoc in the theory group in January 1989. He has been working mainly on problems related to complexity theory.

Together with Babai and Szegedy [33], he proved lower bounds for the multiparty communication complexity of certain simple functions. These bounds were used to obtain a pseudorandom generator for Logspace without relying on any unproven assumptions.

In [245], Nisan obtained a full characterization of the parallel time needed to compute any boolean function on a CREW PRAM in terms of the function's decision tree complexity.

In joint work with Linial [210], the question of obtaining approximate versions of the inclusion-exclusion formula is tackled. Tight upper and lower bounds are proved for several formulations of this question.

Nisan and Kilian [179] considered cryptographic protocols in the setting where all parties are space-bound. In this setting, they design secure protocols for a wide spectrum of cryptographic problems. The security of these protocols is proved without relying on any unproven assumptions.

In his joint work with Linial and Mansour [208], constant depth circuits are studied in terms of their Furier transform. It is shown that almost all of the power spectrum of a function in AC^0 lies in the low coefficients. This fact is used to obtain a learning algorithm for constant depth circuits, as well as several others results.

Marios C. Papaefthymiou

Papaefthymiou began his studies as a graduate student at MIT in September 1988. He is working on his SM thesis under the supervision of Leiserson.

His research focuses on the design of efficient algorithms for pipelining of combinational circuitry. A general framework for this problem has been given by Leiserson and Saxe [204].

Papaefthymiou has given an $O(E)$ optimal algorithm for minimum latency pipelining of combinational circuitry with constrained clock period. He also investigates methods for pipelining combinational circuitry using minimum number of registers.

James K. Park

James K. Park spent most of the last year collaborating with Aggarwal (IBM, Yorktown Heights) and Kravets on a number of problems relating to totally monotone arrays. Park's work with Aggarwal (described in [13][14], and another manuscript "Parallel Searching in Multidimensional Monotone Arrays," currently in preparation) centers on the problem of finding maximum entries in totally monotone arrays and applications of efficient sequential and parallel algorithms for this problem to problems in computational geometry, dynamic programming, string matching, and VLSI river routing. This work generalizes and extends the results of [11]. (Park's Master's thesis [254], finished in January, is also on this subject.) Park's work with Kravets (described in Kravets' Master's thesis [191]) considers two more comparison problems—sorting and computing order statistics—in the context of totally monotone arrays and applications of efficient solutions to these problems.

In the coming year, Park plans to continue his research relating to totally monotone arrays and computational geometry.

Cynthia A. Phillips

Phillips developed an $O(\lg^2 n)$-time $(n + e)/\lg n$-processor deterministic parallel algorithm to contract general n-node, e-edge graphs to a single node. This algorithm is used as a subroutine in an algorithm developed with Leiserson to contract n-node bounded-degree graphs in $O(\lg n + \lg^2 \gamma)$ time with high probability where γ is the maximum genus of any connected component. A deterministic version runs in time $O(\lg n \lg^* n + \lg^2 \gamma)$. The algorithm for bounded-degree graphs uses $n/\lg n$ processors [262]. The contraction algorithm can be used to solve the connected-components, biconnected-components, and spanning-tree problems.

Phillips, with Zenios of the University of Pennsylvania, completed a preliminary experimental study of the solution of large assignment problems on the Connection Machine (TM) multiprocessor. The assignment problem is also known as maximum-weight bipartite matching. They developed heuristics to improve sequential "tail" behavior which seems to limit the usefulness of many current parallel algorithms for the assignment problem and related flow problems [263].

Phillips will be writing her thesis this summer. Among the new research that will probably be included is an analysis of the permutation distribution of the Benes network. In other words, how many distinct ways can the switches of a Benes network be set to yield a given permutation? If the permutations are well distributed, then pseudorandomly setting the switches of a Benes network may yield a good pseudorandom permutation network.

Satish B. Rao

In [184] Koch, Leighton, Maggs, Rao and Rosenberg study the problem of emulating T_G steps of an N_G-node guest network on an N_H-node host network. Although many isolated emulation results have been proved for specific networks in the past, and measures such as dilation and congestion were known to be important, the field has lacked a model within which general results and meaningful lower bounds can be proved. They attempt to provide such a model, along with corresponding general techniques and specific results in this paper.

Leighton and Rao have developed an approximate min-cut max-flow theorem for a type of multicommodity flow problem. This theorem yields an approximation algorithm for finding a separator in arbitrary graphs that costs at most a $O(\log^2 n)$ times the optimal. They also used the theorem to show that any permutation can be routed on an arbitrary network so that the congestion of any edge and path length of any message is within a $O(\log n)$ factor of optimal. In joint work with Maggs, they explore the problem of scheduling messages on paths with given congestion and length so that the routing time is minimized.

Jon G. Riecke

Riecke continues to work in the area of semantics and logic of programming languages, with two primary interests: the semantics of continuations, and the theory of "lazy" (call-by-name) functional languages. Working jointly with Meyer, he investigated some seemingly known—but undocumented—problems in the theory of continuations. More specifically, Meyer and Riecke showed that either programming with continuations explicitly or using special "continuation-accessing" operators (e.g., Scheme's `call/cc`) leads one to conclude different facts about code; old equivalences between programs may no longer hold in a setting with continuations. The implications of these results and their precise statements are reported in [234] and in Riecke's SM thesis [273].

The theory of lazy languages, begun by Abramsky and Ong, has also become a focus of Riecke's work. Lazy functional languages pass arguments by name (that is, arguments are not evaluated before passing), but nevertheless stop evaluating higher-order expressions—functions—when they can build a *closure*. The usual Scott-style semantics do not predict this termination behavior correctly: a divergent functional and a closure that always diverges have the same meaning. Bloom and Riecke [54] developed a model for a *typed* lazy language that accurately reflects the behavior of the interpreter. Cosmadakis and Riecke (in a forthcoming paper) used the model to develop principles for reasoning about lazy programs, and proved that equalities between terms in a fragment of the language are decidable.

In the past year, Riecke has also become interested in intuitionistic logic and type theory, and its applications to the theory of programming languages. He will continue his reading, as well as pursuing previous lines of research.

Phillip Rogaway

Rogaway is a third year graduate student working under Micali. He has been working on cryptography and complexity theory.

Rogaway's Master's thesis evolved into the CRYPTO-88 paper which easily won the award for most coauthors [41]. This paper establishes that an injective one-way function suffices to prove all of **IP** in computational zero-knowledge. It also shows that the "envelope model" for bit commitment suffices to show all of **IP** has perfect zero-knowledge proofs.

Rogaway investigated generalized notions of knowledge complexity, e.g., protocols that release a "small" (but nonzero) amount of information. Recently he has been working on reducing the interaction required for secure distributed computation.

John Rompel

In January, Rompel completed his Master's thesis [277] based on approximation algorithms for graph coloring developed last year with Berger [45].

More recently, Rompel has been working on problems in the field of parallel algorithms. Rompel, together with Berger, [46] developed a general framework for removing randomness from randomized NC algorithms whose analysis uses only polylogarithmic independence. Previously no techniques were known to determinize those RNC algorithms depending on more than constant independence. One application of their techniques is an NC algorithm for the set discrepancy problem, which can be used to obtain many other NC algorithms, including a better NC edge coloring algorithm. As another application of their techniques, they provided an NC algorithm for a hypergraph coloring problem.

Rompel, working with Berger and Shor [47], gave NC approximation algorithms for the unweighted and weighted set cover problems. Their algorithms use a linear number of processors and give a cover that has at most $\log n$ times the optimal size/weight, thus matching the performance of the best sequential algorithms. Previously, there were no known parallel algorithms for the general set cover problem. Berger, Rompel and Shor devised a randomized algorithm, depending on only pairwise independence, and then converted it to a deterministic one. Furthermore, they applied their set cover algorithm to learning theory, giving an NC algorithm to learn the concept class obtained by taking the closure under finite union or finite intersection of any concept class of finite VC-dimension which has an NC hypothesis finder. In addition, they gave a linear-processor NC algorithm for a variant of the set cover problem first proposed by Chazelle and Friedman, and used it to obtain NC algorithms for several problems in computational geometry.

Arie Rudich

Rudich began his first year as a graduate student at MIT in September 1988. He is working on an SM thesis supervised by Meyer on dataflow theory which should be complete by January 1990.

His research aims to generalize recent results by Rabinovich and Trakhtenbrot [269] and by Lynch and Stark [228], which precisely delimit the classes of dataflow networks for which Kahn's "Least Fixed Point Principle" [173] applies, showing that Kahn's Principle fails precisely where Brock-Ackerman-like anomalies [68] begin. Rabinovich and Trakhtenbrot established this boundary without distinguishing completed and incompleted output streams. Rudich aims to show that the results carry over to the more conventional model where the completed/incompleted distinction is maintained.

Robert E. Schapire

Schapire continued to work with Rivest on the problem of inferring an unknown finite-state automaton from its input/output behavior. In [274], they introduce a powerful new technique, based on the inference of homing sequences, for solving this problem in the absence of a means of resetting the machine to a start state. Their inference procedures experiment with the unknown machine, and from time to time require a teacher to supply counterexamples to incorrect conjectures about the structure of the unknown automaton. In this setting, they describe a learning algorithm that, with probability $1 - \delta$, outputs a correct description of the unknown machine in time polynomial in the automaton's size, the length of the longest counterexample, and $\log(1/\delta)$. They present an analogous algorithm that makes use of a diversity-based representation of the finite-state system. Their algorithms are the first that are provably effective for these problems, in the absence of a "reset." They also present probabilistic algorithms for permutation automata which do not require a teacher to supply counterexamples. For inferring a permutation automaton of diversity D, they improve the best previous time bound by roughly a factor of $D^3/\log D$.

In January, Schapire led a team participating in the robot building contest of the AI Lab's "Winter Olympics." The goal of their project was to build a robot capable of performing some simple learning task. In particular, the robot they built, named S'bot (for Smartbot or Spotbot), was able to learn from experience how to avoid running into walls and other obstacles. Their team consisted of Amsterdam, Blum, Goldman, Moore, Rivest, and Schapire.

Schapire has also been working with Goldman and Rivest on the problem of inferring a binary relation [130] between n objects of one kind and m of another. This can be viewed as the problem of inferring an $n \times m$ binary matrix. Their goal has been to minimize the number of prediction mistakes made by a learner presented with such a matrix one entry at a time. They have been able to prove numerous upper and lower mistake bounds for several variations of this problem.

Finally, Schapire has been looking at problems relevant to the distribution-free ("pac") learning model introduced by Valiant [304]. In [281], Schapire considers the problem of improving the accuracy of a hypothesis output by a learning algorithm. He shows that a model of learnability, called *weak learnability*, in which the learner is only required to perform slightly better than guessing, is as strong as a model in which the learner's error can be made arbitrarily small. His result may have significant applications as a tool for efficiently converting a mediocre learning algorithm into one that performs extremely well.

Leonard Schulman

Schulman spent most of his time on coursework this year. In the spring he developed an algorithm for sorting n elements on an n-node ring of processors in the optimal time $n/2$. This requires only constant capacity at each node in the word model. Mansour proved a closely related lower bound and these two results have been combined in a joint paper to be submitted shortly.

During the summer of 1989, Schulman intends to read under the guidance of Sipser.

Eric J. Schwabe

Schwabe has been working on problems involving the efficient implementation of dynamic structures on fixed-connection networks. In particular, he worked with Leighton, Newman, and Ranade (Berkeley) on the problem of dynamically embedding binary trees in butterfly and hypercube networks [201]. Randomized embedding algorithms were found for both networks which simultaneously optimize load (the maximum number of tree nodes mapped to a processor) and dilation (the maximum distance in the network between adjacent tree nodes) for trees which are a logarithmic factor larger than the host network. An improved algorithm for the hypercube was found which optimizes load and dilation for arbitrary binary trees, while also keeping congestion (the number of times a hypercube edge is 'traced over' by an embedded tree edge) low. Also, lower bounds were proved which show that deterministic algorithms cannot simultaneously optimize load and dilation.

Schwabe has also been studying the relative strengths of the butterfly and shuffle-exchange graphs as interconnection networks. He proved that normal hypercube algorithms (those which use only one dimension of hypercube edges at a time, and adjacent dimensions in consecutive time steps) can be simulated on a butterfly network with only a constant slow-down, a result which was previously known only for the shuffle-exchange graph. A version of this result is being prepared for journal submission. In addition, he recently discovered a one-to-one embedding of the butterfly into the shuffle-exchange graph with constant dilation and congestion, and expansion $2^{O(\sqrt{\log N})}$, improving a result of Koch, et. al. [184].

Over the next year, Schwabe plans to work on relating the ideas in [201] to other problems in parallel memory management, and to continue his investigation of the shuffle-exchange vs. the butterfly.

Alan Sherman

Sherman (now faculty at Tufts University) has completed a monograph on the PI System for placement and interconnect of custom VLSI circuits [285]. The PI System was designed and implemented at MIT under the leadership of Rivest; Sherman was one of the key architects. The monograph is being published by Springer-Verlag. Beginning September 1989, Sherman will join the faculty at the University of Maryland, Baltimore County.

Robert Sloan

Sloan's primary area of interest this past year was computational learning theory. His major activity for the year was preparing his doctoral dissertation [290]. Most of the other work in computational learning theory described here is also contained in that work.

Much of his work was within Valiant's model of probably approximately correct learning [304]. Working with Helmbold and Warmuth while visiting the University of California at Santa Cruz, he developed an algorithm for learning certain complex combinations of concept classes known to be learnable [159].

In [275], the problem of learning arbitrary boolean concepts in the Valiant model—by breaking them into pieces and learning one piece at a time is studied. In other work, Sloan studied the effects of different sorts of noise on learning in the Valiant model [288].

He explored an alternate model of inductive inference in [276].

Sloan also remains interested in the subject of cryptography, and spent some time studying different definitions of zero-knowledge [289].

Clifford Stein

Stein has been working with Shmoys on developing parallel algorithms for combinatorial optimization problems. Together with Klein of Harvard, he developed a parallel algorithm to find a maximal set of edge disjoint cycles in an undirected graph in $O(\log n)$ time using m processors on a CRCW PRAM. A maximal set of edge disjoint cycles is a set of cycles whose removal from the graph renders the graph acyclic. Stein and Klein have also been able to generalize this result to multi-graphs and obtain an algorithm which runs in $O(\log n \log C)$ time, where C is the largest multiplicity of any edge [181].

Using this algorithm, Stein has developed an algorithm which finds a *cycle cover* containing $O(m + n \log n)$ edges using $O(\log^2 n)$ time on m processors. A cycle cover is a set of cycles such that every edge in the graph appears in at least one cycle.

Stein has observed that the parallel matching algorithms of [242] and [174] can be combined with scaling to achieve RNC algorithms for the assignment problem which use a number of processors independent of the size of the largest number in the problem, by slowing down the running time by a factor proportional to the logarithm of the size of the largest number in the problem.

Stein has also been rewriting his undergraduate thesis [295] for publication. Together with Ahuja, Orlin, and Tarjan, he developed efficient algorithms for a wide variety of network flow problems in bipartite graphs. The main results are of the following form: given a bipartite graph with n nodes, but only n_1 nodes in the smaller half of the bipartition, an algorithm which runs in time $O(f(n, m))$ can be converted into an algorithm which runs in time $O(f(n_1, m) \cdot n_1 m)$. This approach leads to an algorithm for bipartite maximum flow which runs in $O(n_1 m \log(\frac{n_1^2}{m} + 2))$ time, an algorithm for bipartite minimum cost circulation which runs in $(n_1 m \log n_1 \log(n_1 C))$ time, and an algorithm for parametric maximum flow which solves l bipartite maximum flow problems in $O(ln + n_1 m \log(\frac{ln_1 + n_1^2}{m} + 2))$ time.

Margaret C. Tuttle

Tuttle joined the Theory Group this year and has been working with Shmoys on approximation algorithms for the Mixed Postman Problem: given a weighted graph G, find a least-cost tour of G which traverses each edge at least once. When G is totally directed or totally undirected, the problem can be solved in polynomial time. When G is a mixed graph (i.e., some edges are directed and some are undirected), the problem is NP-complete (as shown by Papadimitriou in 1976).

This summer she will continue working with Shmoys.

Joel Wein

Wein has been working with Shmoys on parallel graph algorithms. He recently extended a result of Karloff's to obtain a Las Vegas *RNC* algorithm for minimum weight perfect matching, where the weights are represented in unary. This problem was shown to be in *RNC* by Karp, Upfal, and Wigderson, but the algorithm was *Monte Carlo* in nature: it yielded a correct solution with high probability, but was unable to determine if the solution was indeed optimal. Wein developed a way to carry out this certification in *RNC*, yielding a robust *Las Vegas* algorithm that can verify optimality. The result utilizes a structure theorem of Sebö for the t-join problem and yields an *RNC* Las Vegas algorithm for that problem as well.

Over the summer, Wein worked at Thinking Machines Corporation, developing practical Connection Machine implementations for various optimization problems. He intends to continue working on both practical and theoretical aspects of parallel computation.

Su-Ming Wu

Working with Tardos, Wu has developed an $O(n^2)$ algorithm for the problem of finding two edge-disjoint paths in a graph [312]. The basis for the algorithm is a graph-theoretic proof of Seymour (Bell Communications Research Laboratory).

13.3 Publications

[1] Y. Afek, B. Awerbuch, and H. Moriel. Overhead of resetting a communication protocol is independent of the size of the network. May 1989. Unpublished manuscript.

[2] A. Agarwal, G. E. Blelloch, R. L. Krawitz, and C. A. Phillips. Four vector-matrix primitives. In *First Symposium on Parallel Algorithms and Architectures*, 1989.

[3] A. Aggarwal, T. Leighton, and K. Palem. Area-time optimal circuits for iterated addition in VLSI. November 1988. Submitted to IEEE Transactions on Computers.

[4] A. Aggarwal and J. Park. Sequential searching in multidimensional monotone arrays. 1989. Submitted for publication.

[5] A. Aggarwal and J. Park. Notes on searching in multidimensional monotone arrays. In *29th Symposium on Foundations of Computer Science*, pages 597–512, IEEE, 1988.

[6] A. Appel and T. Jim. Continuation-passing, closure-passing style. In *16th Symposium on Principles of Programming Languages*, 1989.

[7] B. Awerbuch. Distributed shortest paths algorithms. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, Seattle, WA, pages 230–240, ACM SIGACT, ACM, May 1989.

[8] B. Awerbuch. On the effects of feedback in dynamic network protocols. In *29th Annual Symposium on Foundations of Computer Science*, pages 231–245, IEEE, October 1988.

[9] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Compact distributed data structures for adaptive network routing. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, Seattle, WA, pages 230–240, ACM SIGACT, ACM, May 1989.

[10] B. Awerbuch, A. Goldberg, M. Luby, and S. Plotkin. Network decomposition and locality in distributed computation. 1989. To appear.

[11] B. Awerbuch, O. Goldreich, D. Peleg, and R. Vainish. *On the Message Complexity of Broadcast: Basic Lower Bound.* Technical Memo MIT/LCS/TM-365, MIT Laboratory for Computer Science, July 1988. Also accepted for publication in *Journal of the ACM.*

[12] B. Awerbuch, O. Goldreich, and A. Herzberg. A quantitative approach to dynamic networks. May 1989. Unpublished manuscript.

[13] B. Awerbuch, Y. Mansour, and N. Shavit. Polynomial end-to-end communication. In *30th Annual Symposium on Foundations of Computer Science*, IEEE, 1989. To appear.

[14] B. Awerbuch and M. Sipser. Dynamic networks are as fast as static networks. In *29th Annual Symposium on Foundations of Computer Science*, pages 206 220, IEEE, October 1988.

[15] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols and pseudorandom generators for logspace. In *Proceedings of the 21st STOC Symposium*, ACM, 1989.

[16] F. Barahona and E. Tardos. Note on Weintraub's minimum cost flow algorithm. *SIAM Journal on Computing*, 1989.

[17] D. Beaver and S. Goldwasser. Multi-party computation with faulty majority. March 1989. Unpublished.

[18] M. Bellare and S. Goldwasser. New paradigms for digital signature schemes and message authentication based on noninteractive zero-knowledge proofs. March 1989. Unpublished.

[19] M. Bellare and S. Micali. How to sign given any trapdoor function. In *Proceedings of CRYPTO-88*, Springer-Verlag, 1988.

[20] M. Bellare and S. Micali. Noninteractive oblivious transfer and applications. March 1989. Unpublished.

[21] M. Bellare, S. Micali, and R. Ostrovsky. Parallelizing zero-knowledge proofs and perfect completeness zero-knowledge. April 1989. Unpublished.

[22] S. Ben-david, G. M. Benedek, and Y. Mansour. The passive student is really weaker. In *COLT*, 1989. Submitted for publication.

[23] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proof systems, removing intractability assumptions. In *Proceedings of the 20th* STOC, ACM, 1988.

[24] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Efficient identification schemes using two prover interactive proofs. March 1989. Unpublished.

[25] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway. Everything provable is provable in zero-knowledge. In *Proceedings of CRYPTO-88*, Springer-Verlag, 1988.

[26] B. Berger. *Data Structures for Removing Randomness.* Technical Report MIT/LCS/TR-436, MIT Laboratory for Computer Science, December 1988.

[27] B. Berger and P. Shor. *Tight Bounds for the Acyclic Subgraph Problem.* Technical Report MIT/LCS/TR-443, MIT Laboratory for Computer Science, June 1989. Submitted for publication.

[28] B. Berger and J. Rompel. A better performance guarantee for approximate graph coloring. *Algorithmica*, 1988.

[29] B. Berger and J. Rompel. Simulating $(\log^c n)$-wise independence in nc. In *30th Symposium on Foundations of Computer Science*, IEEE, 1989. To appear. Also Technical Report MIT/LCS/TR-435, MIT Laboratory for Computer Science.

[30] B. Berger, J. Rompel, and P. Shor. Efficient nc algorithms for set cover with applications to learning and geometry. In *30th Symposium on Foundations of Computer Science*, IEEE, 1989. To appear. Also appeared as Technical Report MIT/LCS/TR-444, MIT Laboratory for Computer Science.

[31] B. Berger, M. Brady, D. Brown, and T. Leighton. Nearly optimal algorithms and bounds for multilayer channel routing. February 1989. Submitted to *Journal of the ACM.*

[32] F. Berman, D. Johnson, T. Leighton, P. Shor, and L. Snyder. Generalized planar matching. *Journal of Algorithms*, 1989. To appear.

[33] D. Bertsimas and M. Grigni. On the space-filling curve heuristic for the euclidean traveling salesman problem. *Operations Research Letters*, 1989. To appear.

[34] S. Bhatt, F. Chung, T. Leighton, and A. Rosenberg. Universal graphs for bounded-degree trees and planar graphs. *SIAM Journal Discrete Math*, 1989. To appear.

[35] S. Bhatt, F. Chung, T. Leighton, and A. Rosenberg. Efficient embedding of trees in hypercubes. October 1988. Submitted to *SIAM Journal on Computing.*

[36] S. Bhatt, F. Chung, J. Hong, T. Leighton, and A. Rosenberg. Optimal simulations by butterfly networks. September 1988. Submitted to *Journal of the ACM.*

[37] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced: preliminary report. In *15th Symposium on the Principles of Programming Languages*, pages 229–239, ACM, 1988. Final version in preparation for journal submission.

[38] B. Bloom. Can LCF be topped? In *Proceedings of LICS '88*, 1988. August 1988.

[39] B. Bloom and A. R. Meyer. A remark on the bisimulation of probabilistic processes. In *Logic at Botic '89. Proceedings*, pages 26–40, July 1989.

[40] B. Bloom and A. R. Meyer. Experimenting with process equivalence. January 1989. Twelve page extended abstract, to be submitted.

[41] B. Bloom and J. G. Riecke. LCF should be lifted. In *Proceedings of AMAST*, pages 133–136, 1989.

[42] A. Blum. An $\tilde{O}(n^{0.4})$-approximation algorithm for 3-coloring (and improved approximation algorithms for k-coloring). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, Seattle, WA, May 1989.

[43] A. Blum and R. Rivest. Training a 3-node neural network is NP-complete. In *Advances in Neural Information Processing Systems 1*, pages 494–501, Morgan Kaufmann, 1988. Also presented at the 1988 Workshop on Computational Learning Theory.

[44] A. Blum. *On the Computational Complexity of Training Simple Neural Networks.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised by R.L.Rivest.

[45] M. Blum, P. Feldman, and S. Micali. Proving security against chosen cyphertext attack. In *Proceedings of CRYPTO-88*, Springer-Verlag, 1988.

[46] R. B. Boppana and M. Sipser. The complexity of finite functions. 1989. To appear in the *Handbook of Theoretical Computer Science.*

[47] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37:156–189, 1988.

[48] G. Brassard, C. Crépeau, and M. Yung. Everything in NP can be argued in perfect zero-knowledge in a constant number of rounds. In *16th ICALP*, Springer-Verlag, 1989. To appear.

[49] G. Brassard and C. Crépeau. Sorting out zero-knowledge. In *Advances in Cryptology: Proceedings of Eurocrypt '89*, Springer-Verlag, 1989. To appear.

[50] K. B. Bruce and A. R. Meyer. The semantics of second-order polymorphic lambda calculus. In G. Kahn, D. B. MacQueen, and G. Plotkin, editors, *Semantics of Data Types*, pages 131–144, Springer-Verlag, 1984. Also, to appear in *Information and Computation*, January 1990, coauthored with J.C. Mitchell.

[51] T. Bui, C. Heigham, C. Jones, and T. Leighton. Improving the performance of the kernighan-lin and simulated annealing graph bisection algorithms. In *DAC*, June 1989. To appear.

[52] B. Chor, M. Merritt, and D. B. Shmoys. Simple constant-time consensus protocols in realistic fault models. *Journal of the ACM*, 1989. To appear. An earlier version of this appeared in *Proceedings of the Fourth Symposium on Principles of Distributed Computing.*

[53] E. Coffman and T. Leighton. A provably efficient algorithm for dynamic storage allocation. *Journal of Computer and System Sciences*, 38(1):2–35, February 1989.

[54] E. Coffman, L. Flatto, and T. Leighton. First-fit allocation of queues: tight probabilistic bounds on wasted space. May 1988. Submitted to *First ACM/SIAM Symposium on Discrete Algorithms.*

[55] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* McGraw-Hill/MIT Press, 1989.

[56] C. Crépeau. Verifiable disclosure of secrets and applications. In *Advances in Cryptology: Proceedings of Eurocrypt '89*, Springer-Verlag, 1989. To appear.

[57] C. Crépeau and J. Kilian. Weakening security assumptions and oblivious transfer. In *Advances in Cryptology: Proceedings of Crypto '88*, Springer-Verlag, 1988. To appear.

[58] C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions. In *28th Symposium on Foundations of Computer Science*, pages 42–52, IEEE, 1988.

[59] A. DeSantis, S. Micali, and G. Persiano. Noninteractive zero-knowledge proof-systems with auxiliary language. In *Proceedings of CRYPTO-88*, Springer-Verlag, 1988.

[60] C. Dwork, D. Shmoys, and L. Stockmeyer. Flipping persuasively in constant expected time. *SIAM Journal on Computing*, 1989. To appear.

[61] P. Elias. Zero error capacity under list decoding. *IEEE Transactions on Information Theory*, 34:1070–1074, 1988.

[62] P. Elias. *Error-Correcting Codes for List Decoding.* Technical Report MIT/LCS/TM-381, MIT Laboratory for Computer Science, February 1989. Submitted for publication.

[63] M. D. Ernst. Ml typechecking is not efficient. In *Papers of the MIT ACM Undergraduate Conference*, April 1989.

[64] M. D. Ernst. *Adequate Models for Recursive Program Schemes.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989. Supervised by A.R. Meyer.

[65] L. Finkelstein, D. Kleitman, and T. Leighton. Applying the classification theorem for finite simple groups to minimize pin count in uniform permutation architectures. In *Proceedings of AWOC*, pages 247–256, 1988.

[66] L. Fortnow and M. Sipser. Are there interactive protocols for co-NP languages? *Information Processing Letters*, 28:249–251, 1988.

[67] L. Fortnow and M. Sipser. Probabilistic computation and linear time. In *21st Symposium on Theory of Computing*, ACM, 1989. To appear.

[68] L. Fortnow. *Complexity-theoretic Aspects of Interactive Proof Systems.* PhD thesis, MIT, 1989.

[69] M. Foster and R. I. Greenberg. Lower bounds on the area of finite-state machines. *Information Processing Letters*, 30(1):1–7, January 1989.

[70] A. Frank, T. Nishizeki, N. Saito, H. Suzuki, and E. Tardos. Algorithms for routing around a rectangle. 1989. Submitted for publication.

[71] A. Frank and E. Tardos. An application of submodular flows. *Linear Algebra and its Applications*, 1989. To appear.

[72] J. Fried. *VLSI Processor Design for Communications Networks.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised by C.E. Leiserson. Also appears as an MIT VLSI Memo.

[73] J. Fried and B. Kuszmaul. NAP (no ALU processor): the great communicator. In *Frontiers of Massively Parallel Computation*, 1988. An extended version of this paper has been submitted for publication in the *Journal of Parallel and Distributed Computing.*

[74] J. Fried. A VLSI chip set for burst and ATM switching. In *International Communications Conference*, 1989.

[75] J. Fried, D. Ghosh, and J. Daly. A novel content-addressable memory circuit. *Electronics Letters*, 1989.

[76] J. Fried, E. Daly, T. Lyszarcz, and M. Cooperman. A yield-enhanced crosspoint switch chip using e-beam restructuring. *IEEE Transactions on Solid-State Circuits*, 2, 1989.

[77] J. Fried. Yield modeling using the SPIROS redundancy planner. In *First International Conference on Wafer-scale Integration*, 1989.

[78] J. Fried and P. Kubat. Reliability models for facilities switching. 1989. Submitted to *IEEE Transactions on Reliability.*

[79] A. V. Goldberg, S. Plotkin, D. B. Shmoys, and E. Tardos. Interior-point methods in parallel computation. 1989. Submitted for publication.

[80] A. V. Goldberg, E. Tardos, and R. E. Tarjan. Network flow algorithms. In *Flows, Paths and VLSI-layout*, Springer-Verlag, 1989. To appear.

[81] A. V. Goldberg, S. Plotkin, and E. Tardos. Combinatorial algorithms for the generalized circulation problem. In *29th Symposium on Foundations of Computer Science*, pages 432–443, IEEE, 1988. Submitted for publication.

[82] S. A. Goldman and R. L. Rivest. Mistake bounds and efficient halving algorithms. 1989. Submitted for publication.

[83] S. A. Goldman, R. L. Rivest, and R. E. Schapire. Learning binary relations and total orders. 1989. To appear.

[84] S. A. Goldman. A space efficient greedy triangulation algorithm. *Information Processing Letters*, 1989. To appear in May. Earlier version available as Technical Memo MIT/LCS/TM-366, MIT Laboratory for Computer Science.

[85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *Society for Industrial and Applied Mathematics*, 18:186–208, 1989.

[86] O. Goldreich, A. Herzberg, and Y. Mansour. Source to destination communication in the presence of faults. In *Eighth Annual ACM Symposium on Principles of Distributed Computing*, 1989. To appear.

[87] R. I. Greenberg. *Area-universal Networks.* VLSI Memo 524, Massachusetts Institute of Technology, 1989.

[88] R. I. Greenberg, A. T. Ishii, and A. L. Sangiovanni-Vincentelli. MulCh: A multi-layer channel router using one, two, and three layer partitions. In *ICADD88*, pages 88–91, IEEE Computer Society Press, 1988.

[89] R. I. Greenberg and C. E. Leiserson. A compact layout for the three-dimensional tree of meshes. *Applied Mathematics Letters*, 1(2):171–176, 1988.

[90] M. Grigni and D. Peleg. *Tight Bounds on Minimum Broadcast Networks.* Technical Memo MIT/LCS/TM-374, MIT Laboratory for Computer Science, December 1988.

[91] L. A. Hall and D. B. Shmoys. Jackson's rule: making a good heuristic better. *Mathematics of Operations Research*, 1989. To appear.

[92] L. A. Hall and D. B. Shmoys. Approximation schemes for constrained scheduling problems. 1989. Submitted for publication.

[93] M. D. Hansen. Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems. 1989. Submitted to *Symposium on Foundations of Computer Science.*

[94] J. Hastad, T. Leighton, and M. Newman. Fast computation using faulty hypercubes. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, Seattle, WA, ACM SIGACT, May 1989. To appear.

[95] D. Helmbold, R. Sloan, and M. Warmuth. Learning nested differences of intersections closed concept classes. 1989. Submitted to *COLT '89.*

[96] D. S. Hochbaum and D. B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach. *SIAM Journal on Computing*, 17:539–551, 1988.

[97] A. T. Ishii. *A Digital Model for Level-clocked Circuitry.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1988. Supervised by C.E. Leiserson.

[98] L. A. Jategaonkar. *ML with Extended Pattern Matching and Subtypes.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised by A.R. Meyer. To appear.

[99] L. Jategaonkar and J. C. Mitchell. ML with extended pattern matching and subtypes (preliminary version). In *Symposium on Lisp and Functional Programming*, pages 198–211, ACM, 1988.

[100] J. Kilian. *Randomness in Algorithms and Protocols.* PhD thesis, MIT Department of Mathematics, 1989. Supervised by S. Goldwasser.

[101] J. Kilian. Efficient zero-knowledge proof systems with bounded interaction. Submitted to *Symposium on Foundations of Computer Science*, 1989.

[102] J. Kilian and N. Nisan. Space bounded cryptography. Submitted to *Symposium on Foundations of Computer Science*, 1989.

[103] J. Kilian, S. Kipnis, and C. E. Leiserson. The organization of permutation architectures with bussed interconnections. *IEEE Transactions on Computers*, 1989. To appear. Also appeared as Technical Memo MIT/LCS/TM-379 and VLSI memo 89-500. Earlier version appeared in *Proceedings of the 28th IEEE Annual Symposium on Foundations of Computer Science*, pages 305–315, 1987.

[104] G. A. Kindervater, J. K. Lenstra, and D. B. Shmoys. The parallel complexity of TSP heuristics. *J. Algorithms*, 1989. To appear.

[105] S. Kipnis. *Three Methods for Range Queries in Computational Geometry.* Technical Memo MIT/LCS/TM-388, MIT Laboratory for Computer Science, March 1989.

[106] P. Klein and C. Stein. *A Parallel Algorithm for Eliminating Cycles in Undirected Graphs.* Center for Research in Computing Technology Technical Report TR-01-89, Harvard University, March 1989. Submitted to Information Processing Letters.

[107] R. Koch. *An Analysis of the Performance of Interconnection Networks for Multiprocessor Systems.* PhD thesis, MIT Department of Mathematics, 1989. Supervised by F.T. Leighton.

[108] R. Koch. Increasing the size of a network by a constant factor can increase performance by more than a constant factor. In *29th Symposium on Foundations of Computer Science*, pages 221–230, IEEE, October 1988.

[109] R. Koch, T. Leighton, B. Maggs, S. Rao, and A. Rosenberg. Work-preserving emulations of fixed-connection networks. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, Seattle, WA, May 1989. To appear.

[110] D. Kravets. *Finding Farthest Neighbors in a Convex Polygon and Related Problems.* Technical Report MIT/LCS/TR-437, MIT Laboratory for Computer Science, January 1989.

[111] A. Aggarwal and D. Kravets. A linear time algorithm for finding all farthest neighbors in a convex polygon. *Information Processing Letters*, 31:16–20, 1989.

[112] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Sequencing and scheduling: algorithms and complexity. In S. C. Graves, A. H. G. Rinnooy Kan, and P. Zipkin, editors, *The Handbooks of Operations Research and Management Science, Volume IV: Production Planning and Inventory*, North-Holland, 1989. To appear.

[113] T. Leighton and B. Maggs. Expanders might be practical: fast algorithms for routing around faults in multibutterflies. Submitted for publication.

[114] T. Leighton, B. Maggs, and S. Rao. Universal packet routing algorithms. In *Proceedings of the 29th Symposium on Foundations of Computer Science*, pages 256–271, IEEE, October 1988.

[115] T. Leighton, M. Newman, A. G. Ranade, and E. Schwabe. Dynamic tree embeddings in butterflies and hypercubes. In *First Symposium on Parallel Algorithms and Architectures*, ACM, 1989.

[116] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms (extended abstract). In *29th Symposium on Foundations of Computer Science*, page 42?, IEEE, 1988.

[117] T. Leighton, F. Makedon, and I. Tollis. A $2n-2$ step algorithm for routing in an nxn array with constant-size queues. In *ACM SPAA*, June 1989. To appear.

[118] T. Leighton. A $2d-1$ lower bound for 2-layer knock-knee channel routing. November 1988. Submitted to *SIAM Journal of Discrete Mathematics.*

[119] T. Leighton and P. Shor. Tight bounds for minimax grid matching with applications to the average case analysis of algorithms. *Combinatorica*, 1989. To appear.

[120] T. Leighton, C. Leiserson, and E. Schwabe. *Theory of Parallel and VLSI Computation.* Research Seminar Series MIT/LCS/RSS6, MIT Laboratory for Computer Science, March 1989.

[121] C. Leiserson. VLSI theory and parallel supercomputing. In *Decennial Caltech Conference on VLSI*, pages 5–16, MIT Press, March 1989.

[122] C. Leiserson and B. Maggs. Communication-efficient parallel algorithms for distributed random-access machines. *Algorithmica*, 3:53–77, 1988. An early version appears as "Communication-efficient parallel graph algorithms," in *1986 International Conference on Parallel Processing,* pages 861–868, August 1986 (Received Most Original Paper Award at the conference.).

[123] C. Leiserson and J. Saxe. A mixed-integer linear programming problem which is efficiently solvable. *Journal of Algorithms*, 9:114–128, 1988. An early version appears in *21st Annual Allerton Conference on Communication, Control, and Computing,* October 1983.

[124] J. K. Lenstra, D. B. Shmoys, and E. Tardos. Scheduling unrelated parallel machines. *Mathematical Programming*, 1989. To appear. An earlier version of this appeared in the *28th Symposium on Foundations of Computer Science.*

[125] N. Linial, Y. Mansour, and R. L. Rivest. Results on learnability and the Vapnik-Chervonenkis dimension. In *Proceedings of the 29th Symposium on Foundations of Computer Science*, pages 120–129, October 1988.

[126] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform and learnability. 1989. Submitted for publication.

[127] N. Linial and N. Nisan. Approximate inclusion-exclusion. 1989. Submitted for publication.

[128] Y. Mansour and B. S. P. Tiwari. Lower bounds for computations with the floor operation. In *Proceedings of ICALP 1989*, 1989. To appear.

[129] Y. Mansour and B. Schieber. The intractability of bounded protocols for non-fifo channels. In *Eight Annual ACM Symposium on Principles of Distributed Computing*, 1989. To appear.

[130] Y. Mansour, B. Schieber, and P. Tiwari. The complexity of approximating the square root. In *30^{th} Symposium on Foundations of Computer Science*, 1989. To appear.

[131] A. R. Meyer and J. G. Riecke. Continuations may be unreasonable. In *Proceedings of Conference on Lisp and Functional Programming*, pages 63–71, ACM, 1988.

[132] A. R. Meyer. Semantical paradigms· notes for an invited lecture, with two appendices by S. Cosmadakis. In *Third Symposium on Logic in Computer Science*, pages 236–253, IEEE, 1988.

[133] A. R. Meyer and M. A. Taitslin, editors. *Logic at Botic,'89: Proceedings of a Symposium on Logical Foundations of Computer Science.* Volume 363 of *Lecture Notes in Computer Science*, Springer-Verlag, July 1989.

[134] S. Micali and C. P. Schnorr. Super-efficient, perfect random number generators. In *Proceedings of CRYPTO-88*, Springer-Verlag, 1988.

[135] S. Micali and A. Shamir. An improvement of the Fiat-Shamir identification. In *Proceedings of CRYPTO-88*, Springer-Verlag, 1988.

[136] S. Micali and R. Ostrovsky. Simple noninteractive zero-knowledge proofs with preprocessing oblivious transfer. September 1988. To appear.

[137] N. Nisan. Crew prams and decision trees. In *Proceedings of the 21^{th} STOC Symposium*, ACM, 1989.

[138] J. Park. *Notes on Searching in Multidimensional Monotone Arrays.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, January 1989. Supervised by C. E. Leiserson.

[139] C. Phillips. Parallel graph contraction. In *First Symposium on Parallel Algorithms and Architectures*, ACM, 1989. To appear.

[140] C. Phillips and S. A. Zenios. Experiences with large scale network optimization on the Connection Machine. In *Impact of Recent Computer Advances on Operations Research*, Elsevier Science Publishing, 1989.

[141] S. Plotkin and E. Tardos. Improved dual network simplex. 1989. Submitted for publication.

[142] S. Rao. Finding near optimal separators in planar graphs. In *28^{th} Symposium on Foundations of Computer Science*, page 225, IEEE, 1987.

[143] J. G. Riecke. *Should a Function Continue?* Master's thesis, MIT Department of Electrical Engineering and Computer Science, January 1989. Supervised by A.R. Meyer.

[144] R. L. Rivest and R. E. Schapire. Inference of finite automata using homing sequences. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, Seattle, WA, May 1989. To appear.

[145] R. L. Rivest and R. Sloan. A new model for inductive inference. In M. Vardi, editor, *Proceedings of the Second Annual Theoretical Aspects of Reasoning about Knowledge Conference*, pages 13–27, Morgan Kaufmann, March 1988. Submitted to *Information and Computation.*

[146] R. L. Rivest and R. Sloan. Learning complicated concepts reliably and usefully. In *Proceedings AAAI-88*, pages 635–640, American Association for Artificial Intelligence, August 1988.

[147] J. Rompel. *A Better Performance Guarantee for Approximate Graph Coloring.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised by T. Leighton.

[148] R. E. Schapire. The strength of weak learnability. 1989. Submitted for publication.

[149] A. T. Sherman. *VLSI Placement and Routing: The PI Project.* Springer-Verlag, 1989.

[150] D. B. Shmoys and D. Williamson. Analyzing the Held-Karp TSP bound: a monotonicity property with application. 1989. Submitted to *Information Processing Letters.*

[151] D. B. Shmoys and E. Tardos. Computational complexity. In R. Graham, M. Grötschel, and L. Lovász, editors, *The Handbook of Combinatorics*, North-Holland, 1989. To appear.

[152] R. Sloan. *All Zero-Knowledge Proofs are Proofs of Language Membership.* Technical Memo MIT/LCS/TM-385, MIT Laboratory for Computer Science, February 1989.

[153] R. Sloan. *Computational Learning Theory: New Models and Algorithms.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised by R. L. Rivest.

[154] R. Sloan. Types of noise in data for concept learning. In *First Workshop on Computational Learning Theory*, pages 91–96, Morgan Kaufmann, 1988.

[155] E. Tardos. An intersection theorem for supermatroids. *Journal of Combinatorial Theory, B*, 1989. To appear.

[156] S. Wu. An efficient algorithm for finding two edge-disjoint paths in a graph. 1988. Submitted for publication.

Talks

[1] B. Bloom. LCF can't be topped. Lecture given at LICS 1988, June 1988.

[2] G. Brassard, C. Crépeau, and M. Yung. Everything in NP can be argued in perfect zero-knowledge in a constant number of rounds. Lecture given at Eurocrypt, April 1989.

[3] G. Brassard and C. Crépeau. Sorting out zero-knowledge. Lecture given at Eurocrypt, April 1989.

[4] G. Brassard, D. Chaum, C. Crépeau, and I. Damgaard. Conversations that don't say much! Lecture given at McGill University, November 1988.

[5] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. Lecture given at SIAM meetings on Discrete Mathematics, June 1988.

[6] C. Crépeau and J. Kilian. Cryptographic protocols based on nature's random sources. Lecture given at IBM Almaden Research Center, August 1988.

[7] C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions. Lecture given at 27^{th} Foundations of Computer Science, October 1988.

[8] C. Crépeau. From photons to secret computations. Lecture given at Aarhus University, April 1989.

[9] C. Crépeau. Verifiable disclosure of secrets and applications. Lecture given at Eurocrypt, April 1989.

[10] M. D. Ernst. Polymorphic typechecking is exponential. Lecture given at Massachusetts Institute of Technology, April 1989.

[11] J. Fried. Broadband module design: cost/performance tradeoffs. Lecture given at International Workshop on Physical Design of Broadband Switching and Multiplexing Equipment, April 1989.

[12] S. A. Goldman. Learning binary relations and total orders. Lecture given at Center for Intelligent Control Systems Machine Learning Workshop, May 1989.

[13] R. I. Greenberg. MulCh: A multi-layer channel router using one, two, and three layer partitions. Lecture given at Massachusetts Institute of Technology, May 1989.

[14] R. I. Greenberg. Efficient multi-layer channel routing. Lecture given at Georgia Institute of Technology and University of Maryland, March–April 1989.

[15] R. I. Greenberg. Area-universal networks. Lecture given at Polytechnic University, Princeton University, and University of Southern California, February–March 1989.

[16] A. T. Ishii. MulCh: A multi-layer channel router using one, two, and three layer partitions. Lecture given at ICADD88, November 1988.

[17] L. Jategaonkar. Ml with extended pattern matching and subtypes. Lecture given at New York University and IBM Research, September 1988.

[18] J. Kilian. Theory and practice of cryptographic primitives. Lecture given at University of California, Berkeley, and Stanford University, April 1989.

[19] T. Leighton. Survey talk on networks, parallel computation and VLSI design. Lecture given at Trento School on VLSI Computation; ICALP (July); NCUBE and University of Oregon (January); Dartmouth (May), 1989.

[20] T. Leighton. Survey talk on packet routing algorithms. Lecture given at IDA SRC (June 1989), U. British Columbia Distinguished Lecture Series, Stanford (December 1988), ICSI Berkeley, IBM Almaden (January 1989), MIT Center for Intelligent Control (March 1989), DARPA Contractors Meeting; NSF Industry-University Symposium (April 1989), DIMACS Symposium Invited Lecture (May 1989).

[21] T. Leighton. Dynamic tree embeddings in butterflies and hypercubes. Lecture given at ICSI Berkeley, January 1989.

[22] T. Leighton. Fast computation using faulty hypercubes. Lecture given at ACM STOC, May 1989.

[23] T. Leighton. Flows, paths and VLSI layout. Lecture given at Bonn Workshop on Flows, Paths and VLSI Layout; and AWOC, June 1988.

[24] T. Leighton, M. Newman, A. G. Ranade, and E. Schwabe. Dynamic tree embeddings in butterflies and hypercubes. Lecture given at MIT VLSI Research Review, May 1989.

[25] C. E. Leiserson. Very large scale computing. Lecture given at MIT Project MAC 25th Anniversary Symposium, MIT LCS, October 1988.

[26] C. E. Leiserson. VLSI theory and parallel supercomputing. Lecture given at Decennial Caltech Conference on VLSI, California Institute of Technology (March); Thinking Machines Corporation (April), 1989.

[27] B. Maggs. Universal packet routing algorithms. Lecture given at IBM Thomas J. Watson Research Center, April 1989.

[28] A. R. Meyer. An ultimate "Kahn Principle" for dataflow semantics. Lecture given at IBM Research Lab, Distinguished Lecture (January); University of Maryland (February), 1989.

[29] A. R. Meyer. Observing concurrent processes: dataflow. Lecture given at MIT, Project MAC 25th Anniversary Symposium, October 1988.

[30] A. R. Meyer. Semantical paradigms. Lecture given at Third IEEE Symposium on Logic in Computer Science, Invited Lecture (July); Mitre Corporation (December), 1988.

[31] J. Park. Notes on searching in multidimensional monotone arrays. Lecture given at 29th Symposium on Foundations of Computer Science, October 1988.

[32] J. G. Riecke. Observing termination in Scott-style semantics. Lecture given at IBM Research, November 1988.

[33] R. L. Rivest. Learning theory: what's easy and what's hard. Lecture given at MIT, October 1989.

[34] R. L. Rivest. Inference of finite automata using homing sequences. Lecture given at Boston University, March 1989.

[35] R. E. Schapire. Inference of finite automata using homing sequences. Lecture given at 21st Annual Symposium on Theory of Computing, May 1989.

[36] R. E. Schapire. The strength of weak learnability. Lecture given at Northeastern University, Center for Intelligent Control Systems Machine Learning Workshop, 1989.

[37] R. E. Schapire. Diversity-based inference of finite automata. Lecture given at GTE Laboratories, American Control Conference, 1988.

[38] D. B. Shmoys. Jackson's rule: making a good heuristic better. Lecture given at CWI, EURO/TIMS, 1988.

[39] D. B. Shmoys. Using linear programming in the design and analysis of approximation algorithms. Lecture given at Princeton (DIMACS Theory Day), Stanford, and Columbia, 1989.

[40] D. B. Shmoys. Approximation schemes for constrained scheduling problems. Lecture given at Stanford, Oberwolfach, and Cornell, 1989.

[41] C. Stein. Improved algorithms for bipartite network flow. Lecture given at MIT Laboratory for Computer Science, April 1989.

[42] E. Tardos. Recent advances in the theory of network flow algorithms. Lecture given at Summer school on "Paths, Flows and VLSI-layout" at the Operations Research Institute, Bonn F.R.G., 1988.

[43] E. Tardos. Combinatorial algorithms for the generalized circulation problem. Lecture given at University of Waterloo, Mathematical Programming Symposium in Tokyo, Rutgers University, Stanford, Cornell University, SIAM Workshop on Optimization, 1988.

Theory of Distributed Systems

Academic Staff

N. Lynch, Group Leader

Research Staff

A. Fekete	H. Attiya

Graduate Students

K. Goldman	N. Shavit
J. Leo	G. Troxel
S. Ponzio	M. Tuttle

Undergraduate Students

C. Colby	M. Nour

Support Staff

A. Wiseman	L. Sprung

14.1 Introduction

The Theory of Distributed Systems Group has continued its work on algorithms and impossibility results for distributed problems, as well as its work on modeling, proof techniques, and applications. Particular highlights this year include our work on atomic registers, on real time systems, and on the design of a system for simulating distributed algorithms.

14.2 Faculty Reports

Nancy A. Lynch

This year, Nancy Lynch worked on combinatorial results (and modeling) for asynchronous communication protocols. The paper [222] shows the impossibility of implementing reliable data link behavior in the face of certain assumptions about physical channels and about node failures. Besides the combinatorial results, another contribution of this paper is the style of problem specification, done in terms of I/O behavior of physical channels and data links. She has continued this work by proving a related impossibility result for "oblivious" non-FIFO physical channels, and by simplifying the specifications used in [222]. Both of these efforts are still in progress.

Other combinatorial work this year includes some new complexity results for real time computing (see below). Lynch completed revisions of two older papers [105] [69]. Also, she worked on the problem of processor renaming in an asynchronous systems, mostly unsuccessfully.

Lynch also worked on general models for concurrent systems. With Mark Tuttle, she wrote a short paper [226] introducing the I/O automaton model; a longer journal version is still planned. This model is used to redo a pair of prior results, done originally in other models [108][17]; the revisions appear to be somewhat simpler than the originals. Lynch also supervised Magda Nour's SB thesis project, in which Magda established interesting connections between the Unity model of Chandy and Misra and the I/O automaton model. Other work on modeling includes work on modeling real time systems (see below).

In addition, Lynch worked on using I/O automata to verify correctness of complicated concurrent algorithms. This verification work includes the correctness proof [308] of the Gallager, et al. Minimum Spanning Tree algorithm [122], and the paper [309] on Drinking Philosophers. Lynch supervised some revisions of Russel Schaffer's SB thesis on verifying atomic register protocols [280]. She also supervised Chris Colby's SB thesis on verifying the correctness of the Peterson-Fischer tournament-structured mutual exclusion algorithm. Finally, she used I/O automata in her consulting work at Apollo Computer, to verify the correctness of a complex algorithm for managing highly available replicated data. This proof has an interesting structure, based on multivalued abstraction mappings: first, a non-garbage-collected version of the algorithm is proved correct, and then the "real" algorithm, using garbage-collection of old updates, is proved correct using abstraction mappings relating it to the non-garbage-collected version.

Many of these proofs are done in a style and at a level of detail that make them suitable candidates for machine verification; with John Guttag and Steve Garland, Lynch is exploring the possibility of using LP to perform such verification.

Work also continued on the theory of atomic transactions, although at a slower pace than last year. This work is a series of papers on modeling and verifying different kinds of transaction-processing algorithms, culminating in a book [224] to tie them all together. This year, she carried out substantial revisions of [104] and [160] (not yet complete). Our papers on locking algorithms [104] and timestamp algorithms [18] appeared in conferences. Besides the revisions, our current work in progress includes results relating our theorems to those of the "classical theory" of database concurrency control, and results about algorithms that carry out concurrency control simultaneously at several different levels of data abstraction.

Lynch began new work on a theory for real time systems (and more generally, for timing-based systems) as part of the ONR's new initiative on real time computing. She gave a talk at the ONR Workshop on Real Time Systems in November, on "Modeling Real Time Systems." This introductory talk showed how I/O automata, extended to include time as in [238], could be used to model the timing restrictions and requirements of real time computing systems. Working with Hagit Attiya, she continued to pursue some of the ideas in the talk, in particular, to study upper and lower bounds for combinatorial problems in real time systems. For example, in [23], they proved upper and lower bounds on both centralized and distributed versions of a timing-based variant of the mutual exclusion problem (which appeared in the real time literature as the "nuclear reactor problem"). Stephen Ponzio carried out related work (described below) on the Dining Philosophers problem.

Our correctness proofs for the algorithms in [23] turned out to have a very interesting style, adapting standard proof techniques for proving safety properties (such as invariant assertions and abstraction mappings) for use in timing-based systems. We are currently working on developing these proof methods for timing-based algorithms in appropriate generality.

Lynch gave the Keynote Address at the 1988 Symposium on Principles of Distributed Computing. The talk she gave was a survey of the many impossibility results that have been proved in this research field. Preparing for this talk was itself a major project of hunting down the results and classifying them. She has written a paper based on this talk [219], to appear in this year's PODC proceedings.

Also, Lynch put the final touches on the paper [194], which is a survey of the theory of distributed computing research area, and has recently written a new NSF proposal with Baruch Awerbuch.

Research service activity this year included serving as editor of a special issue of IEEE Transactions on Computer Systems on parallel and distributed algorithms, and also as an editor for *Information and Computation.* Lynch also served on the selection committee for this year's ACM Thesis Prize, and on the Program Committee for the annual ACM Symposium on Theory of Computing.

Besides supervising her own research students, Lynch served as reader on thesis committees for Radia Perlman at MIT, and for Lisa Higham at the University of British Columbia.

With Ken Goldman, Lynch put together a set of course notes for her class on Distributed Algorithms [221].

Plans for the near future are to continue her work on combinatorial results, especially those involving timing-based computation, consensus and atomic objects. She also plans to finish her book on atomic transactions and to continue her work on modeling concurrent systems, including those that use timing assumptions and randomization. She will also continue to try to use the I/O automaton model to describe the semantics of other frameworks and languages for concurrent computation, and to prove correctness of difficult concurrent algorithms.

14.3 Research Associate and Student Reports

Hagit Attiya

Research Associate Hagit Attiya worked with Michael Fischer, Da-Wei Wang, and Lenore Zuck, of Yale University, on the Sequence Transmission Problem [22]. Here, a processor should transfer a sequence of data items to another processor over an asynchronous channel. It was shown that there is a protocol using finite-sized messages for this problem over an asynchronous channel that may reorder and delete messages, using finite-sized messages. This is in contrast with the results in [222], where it was shown that there is no *bounded* protocol for solving the related *data link layer problem.*

The rest of the research done by Attiya during this period can be divided into two areas: *timing-based problems*, and *wait-free coordination.*

Timing-based Problems

This work, joint with Nancy Lynch, uses the *timed I/O automata* framework (introduced by [238], see also [218]).

They considered a timing-based variant of the *mutual exclusion* problem. In this variant, only an upper-bound on the time it takes to release the resource is known, as opposed to receiving an explicit signal when the resource is released. Furthermore, the only mechanism to measure real time is an inaccurate clock, whose tick intervals take time between two known constant bounds. Upper and lower bounds on the response time of any algorithm solving this problem were proven, for algorithms where the control is either centralized or distributed.

The lower bound proofs make use of new techniques. In order to prove the correctness of these algorithms, Lynch and Attiya developed a way to transform any timed automaton into a "regular" automaton, by building timing information into the automaton state, thus enabling the use of classic invariant assertion proof technique. Surprisingly, this method can be extended to prove the *performance* of algorithms. This direction is explored in [220].

Wait-free Coordination

Hagit worked with Danny Dolev (of IBM Almaden and Hebrew University) and Nir Shavit on wait-free solutions problems using bounded shared memory. The goal of this research is the development of tools that will eventually enable us to reveal the exact relationship between boundedness of memory and wait-freeness of operations. Two specific problems considered are:

1. *Randomized Consensus*: All the algorithms known for this problem, and in particular the only polynomial algorithm ([19]), use shared memory with unbounded values. Attiya, Dolev, and Shavit found a bounded polynomial solution to the problem [21], answering the open question of Abrahamson [2].
2. *Snapshot Scan*: A *snapshot scan* returns an "instantaneous" picture of memory. Such an algorithm will greatly simplify proofs of concurrent programs, and is an important building block in many algorithms [95][2][19]. The correctness of a bounded construction that solves this problem is currently being proven (joint work with Michael Merritt and Yehuda Afek of AT&T Bell Labs).

In joint work with Mark Tuttle [24], new proof techniques were developed to prove lower bounds for problems in both the shared memory and message passing models of computation. These techniques are nontrivial generalizations of [108][217][67], and, as a result of the similarity of the proofs in the two models, have the advantage of exposing similarities between the shared memory and message passing models. Using these techniques, it is possible to obtain a new tight lower bound for the slotted ℓ-exclusion problem [20] (a similar lower bound was proved for the related problem of ℓ-assignment in [70]). Furthermore, these techniques give simple proofs for known results, such as consensus [108] and processor renaming [20],

Chris Colby

Chris Colby worked on two projects.

Ken Goldman is developing an I/O automaton distributed simulation system. It will be used to aid in the design and study of distributed algorithms using the I/O automaton model. During the summer of 1988, Colby worked on the development of a graphical user-interface for the system. The interface allows the user to graphically configure I/O automaton systems by composing automata hierarchically and specifying topology information. The interface will be used to observe automaton states during simulation, and may eventually be used to guide the particular execution path taken by the simulator.

Colby also finished his undergraduate thesis entitled *Correctness Proofs of the Peterson-Fischer Mutual Exclusion Algorithms*. In this thesis, The Peterson-Fischer 2-process mutual exclusion algorithm [260] is introduced in a slightly modified form. An invariant-assertional proof of mutual exclusion is presented for the 2-process algorithm. Next, the Peterson-Fischer n-process mutual exclusion algorithm is introduced conceptually as a *tournament* of $\lceil \lg n \rceil$ 2-process competitions. A mutual-exclusion proof of the n-process algorithm is presented, based on a mapping between states of the n-process system and states of the 2-process system. This mapping delineates the correspondence between the 2-process code and one iteration (competition) of the n-process code. In this way, the statement of correctness of the 2-process algorithm is used as a lemma for the n-process proof.

Alan Fekete

Alan Fekete was a member of the TDS Group until September 30, 1988. He worked on the theory of concurrency control for nested transactions. In particular, he developed with Nancy Lynch a way to use the general theory given in the paper [223] to prove a sufficient condition for correctness that resembles the "absence of cycles" condition used in the conventional theory of serializability for transactions without nesting. With this condition, they gave a simple direct proof of the correctness of Moss' algorithm for read/update locking. He also found a way to model and verify some "optimistic" timestamp-based concurrency control algorithms, that allow some transactions to proceed in the hope that no errors will occur, and only check that in fact nothing went wrong before commit occurs (rather than before each access to objects). Another area of Fekete's work was the possibility and impossibility of solving certain problems concerned with building a reliable message service on top of an unreliable service. Results proved by Lynch, Mansour, and Fekete [222] were compared with those of Attiya, Fischer, Wang, and Zuck which used a significantly different model of a system, in order to see in which respects the results in one model carried over into the other. A result prove with Lynch shows that some sequence information was essential, even under very weak constraints on the system.

Ken Goldman

Ken Goldman has been working on his Ph.D. thesis, *Simulation of Concurrent Algorithms Using I/O Automata.* He has been designing a strongly typed language based on the I/O automaton model and a simulation system for studying algorithms expressed in that language. The system is intended as a research tool to aid in the design and understanding of distributed algorithms. Chris Colby has been implementing a graphical interface for the system (see above). In [126], Goldman explored the possibility of distributing the simulation in a highly concurrent manner, while fully preserving the semantics of the model.

As part of his area examination, Goldman studied the languages Lisp, Connection Machine Lisp, and Paralation Lisp in terms of their utility for writing efficient scientific programs on the Connection Machine. In [127], he reviews those languages and proposes a set of extensions to Paralation Lisp for improving both the expressiveness and the efficiency of that language.

John Leo

John Leo has been continuing work on his Master's thesis *I/O Automata Techniques and Examples,* expected to be completed by August 1989. The thesis is based on two examples, SIFT and OLYMPIC TORCH, both involving process creation. The emphasis of the thesis is to demonstrate proof techniques and create new tools for correctness proofs using I/O automata. One such tool is a version of mappings from components of a composition to a single higher level specification, adapted from [227]. Other tools will be theorems concerning message passing systems and local improvements.

Magda Nour

Magda Nour worked on and finished her undergraduate thesis entitled *An Automata-Theoretic Model for UNITY.*

UNITY—Unbounded Nondeterministic Iterative Transformation—is a computational model and a proof system to aid in the design of parallel programs developed by K. Mani Chandy and Jayadev Misra at the University of Texas.

The Input/Output Automaton model is a computational model developed by Nancy Lynch and Mark Tuttle that may be used to model concurrent and distributed systems.

This thesis connects these two theories. Specifically, it:

1. defines Unity Automata, a subset of I/O automata based on the UNITY computational model, [illegible] UNITY program;
2. defines a mapping from UNITY programs to UNITY automata;
3. adapts the UNITY proof concepts to the I/O automaton computational model in order to obtain UNITY style proof rules for I/O automata;
4. adapts UNITY composition operators to the I/O automaton model and obtains composition proof rules for them; and
5. considers various examples illustrating the above work.

In addition, this work introduces an augmentation to the I/O automaton model which facilitates reasoning about randomized algorithms, adapts UNITY concepts to it, and presents an example of a UNITY style high probability proof using such a model.

Stephen Ponzio

Stephen Ponzio is currently studying the complexity of solutions to the "dining philosophers problem" in a real time system. If the maximum amount of real time that any processor spends in the critical region is bounded by some constant c, then a simple alternating algorithm guarantees a waiting time of $3c + O(1)$. He shows that in a system of n processors which issue requests asynchronously, no algorithm can guarantee a waiting time of less than $2c$. He also gives an algorithm that guarantees $2c + O(n)$ for n even. Future research will improve upon the known algorithms or raise the lower bound by including terms such as message delay time. Other problems fundamental to distributed computing will also be considered.

Nir Shavit

Nir Shavit worked with Danny Dolev of IBM ARC on the Bounded-Concurrent-Time-Stamping problem. Concurrent time stamping is at the heart of solutions to some of the most fundamental problems in distributed computing. Based on concurrent-time-stamp-systems, elegant and simple solutions to core problems such as fcfs-mutual-exclusion, construction of a multi-reader-multi-writer atomic register, probabilistic consensus, etc. were developed. Unfortunately, the only known implementation of a concurrent-time-stamp-system has been theoretically unsatisfying, since it requires unbounded size time-stamps; in other words, unbounded memory. Not knowing if bounded concurrent-time-stamp-systems are at all constructible, researchers were led to constructing complicated problem-specific solutions to replace the simple unbounded ones. In this work, for the first time, a bounded implementation of a concurrent-time-stamp-system is presented. It provides a modular unbounded-to-bounded transformation of the simple unbounded solutions to problems such as above. It allows solutions to two formerly open problems, the bounded-probabilistic-consensus problem of Abrahamson [2], and the FIFO-l-exclusion problem of [110], and a more efficient construction of mrmw-atomic registers. This work [95] was presented at STOC 1989.

Shavit also worked with Hagit Attiya and Danny Dolev on wait-free solutions of problems using bounded shared memory. The goal of this research is the development of tools that will eventually enable us to reveal the exact relationship between unboundedness of memory and wait-freeness of operations. Two specific problems we considered:

1. *Randomized Consensus*: All the algorithms known for this problem, and in particular the only polynomial algorithm (due to Aspens and Herlihy), use shared memory with unbounded values. A bounded polynomial solution to the problem [21], answering the open question of Abrahamson [2], will be presented at POCD 1989.

2. *Snapshot Scan*: We are interested in a scan algorithm that will return an "instantaneous" picture of memory. Such an algorithm will greatly simplify proofs of concurrent programs, and is an important building block in many algorithms [95][2][19]. We are currently in the process of proving the correctness of a bounded construction that we believe solves this problem (joint work with Mike Merritt and Yehuda Afek) .

Shavit also worked with Baruch Awerbuch and Yishay Mansour on the end-to-end problem in unreliable networks. This is a fundamental problem, dealing with the question of how to assure that two nodes in a communication network such as Bitnet, can guarantee communication even if they are only *eventually connected* (see [315]). We present the first polynomial solution to the problem [31], opening the possibility that with further research, a practically efficient solution to the problem may be found.

Shavit is currently working with Hagit Attiya and Nancy Lynch on a line of research that will lead to a better understanding of the notion of wait-freeness and its relation to fault tolerance. We are currently attempting to verify if existing definitions really capture the intuitive properties attributed to wait-free primitives.

Shavit is in the process of completing a draft of joint work with Mike Merritt and Yehuda Afek on the local snapshot algorithm, an algorithm that performs snapshots in distributed message passing systems, with time and communication complexity dependent on the flow of computation of the application, rather than the size of the complete network.

Also, Shavit is writing his Ph.D. thesis, touching on many of the research topics mentioned above.

Greg Troxel

Greg Troxel refined an algorithm he developed for detecting and recovering from process-execution resource deadlock in a system using remote procedure calls intended for the Fault Tolerant Parallel Processor being developed at the Charles Stark Draper Laboratory. He is constructing a proof using I/O automata that this algorithm is correct. Interesting issues in the proof are the formal specification of correctness conditions of the algorithm, since this algorithm functions as a scheduler for the remote procedure call system, and demonstration of liveness properties.

Mark Tuttle

Mark Tuttle is primarily interested in understanding the correctness and construction of distributed algorithms in terms of the "knowledge" individual processors in a distributed system have about their environment (e.g., the local states of neighboring processors, etc.).

His current interest is understanding cryptographic protocols and system security in terms of formal notions of knowledge (e.g., [145]). In order to think about probabilistic protocols like these in terms of knowledge, we have to be able to answer the question "What should it mean for an agent to know or believe an assertion is true with probability .99?". Different papers [102][109][145] give different answers, choosing to use quite different probability spaces when computing the probability an agent assigns to an event. In [146], joint work with Joe Halpern, they show no single choice is correct in all contexts, and show for any given context how to make the most appropriate choice. They show that each choice can be understood in terms of a betting game, and that each choice corresponds to betting against a different opponent. They consider three types of adversaries. The first selects the outcome of all nondeterministic choices in the system; the second represents the knowledge of the agent's opponent (this is the key place the above-mentioned papers differ); the third is needed in asynchronous systems to choose the time the bet is placed. They illustrate the need for considering all three types of adversaries with a number of examples. Given a class of adversaries, they show how to assign probability spaces to agents in a way most appropriate for that class, where "most appropriate" is made precise in terms of this betting game. They conclude by showing how different assignments of probability spaces (corresponding to different opponents) yield different levels of guarantees in coordinated attack.

In [241], it is shown how to construct extremely fast, efficient protocols for problems like consensus in synchronous systems, problems requiring processors to perform coordinated actions simultaneously. It is shown that the construction of such protocols reduces to testing for a state of knowledge called *common knowledge*. Unfortunately, these results do not extend to asynchronous systems; in fact, it is known the state of common knowledge cannot be attained in such systems [144]. In such systems, however, the state of *eventual common knowledge* [144] appears to be very closely related to the solution of such problems, but there are no useful tools for proving that this state of knowledge is or is not attained, let alone for a processor to test for this state of knowledge. In [302], Tuttle gives a new game-theoretic characterization of eventual common knowledge, a characterization that may be a first step in developing such tools.

In joint work with Hagit Attiya [24], new proof techniques are developed to prove lower bounds for problems in both the shared memory and message passing models of computation. These techniques are nontrivial generalizations of [108][217][67], and, as a result of the similarity of the proofs in the two models, have the advantage of exposing similarities between the shared memory and message passing models. Using these techniques it is possible to obtain easily known lower bounds on consensus [108], processor renaming [20], and ℓ-exclusion [70].

Other work by Tuttle this year includes further exposition [226] with Nancy Lynch of the Input/Output Automaton model of distributed computation formalized in the course of his Master's thesis work [225], and further thought [238] on the problem of adding time to the I/O automaton model to allow the model to be used to reason about real time systems.

14.4 Publications

[1] J. Aspnes, A. Fekete, N. Lynch, M. Merritt, and W. Weihl. A theory of timestamp-based concurrency control for nested transactions. In *Proceedings of 14th International Conference on Very Large Data Bases*, pages 431–444, Los Angeles, CA, August 1988.

[2] H. Attiya, D. Dolev, and N. Shavit. Bounded polynomial randomized consensus. To appear in *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*, 1989.

[3] H. Attiya, M. Fischer, D. Wang, and L. Zuck. Reliable communication over an unreliable channel. In progress.

[4] H. Attiya and N. Lynch. Time bounds for real time process control in the presence of timing uncertainty. Submitted for publication.

[5] H. Attiya and M. Snir. Better computing on an anonymous ring. Submitted for publication.

[6] H. Attiya and M. Tuttle. Bounds for slotted ℓ-exclusion. February 1989. Unpublished manuscript.

[7] J. Burns and N. Lynch. Mutual exclusion using indivisible reads and writes. In *Proceedings of 18th Annual Allerton Conference on Communications, Control, and Computing*, pages 833–842, 1980. Revision in progress.

[8] A. Fekete, N. Lynch, M. Merritt, and W. Weihl. Commutativity-based locking for nested transactions. In *Proceedings of Third International Workshop on Persistent Object Systems*, pages 113–127, Newcastle, Australia, January 1989.

[9] M. Fischer, N. Lynch, J. Burns, and A. Borodin. Distributed FIFO allocation of identical resources using small shared space. *ACM Transactions on Programming Languages and Systems*, 11(1):90–114, January 1989.

[10] K. Goldman. Highly concurrent logically synchronous multicast. Submitted for publication.

[11] K. Goldman. *Paralation Views: Abstractions for Efficient Scientific Computing on the Connection Machine.* To appear as a Technical Report.

[12] J. Halpern and M. Tuttle. Probabilistic knowledge and the power of the adversary. January 1989. Submitted for publication.

[13] L. Lamport and N. Lynch. Distributed computing. Chapter of *Handbook on Theoretical Computer Science*. Also, to be published as Technical Memo MIT/LCS/TM-384, MIT Laboratory for Computer Science, 1989.

[14] N. Lynch. Modelling real time systems. November 1988.

[15] N. Lynch. A hundred impossibility proofs for distributed computing. Submitted to PODC 89.

[16] N. Lynch and H. Attiya. Assertional proofs of timing properties. In progress.

[17] N. Lynch and K. Goldman. *Distributed Algorithms.* Research Seminar Series MIT/LCS/RSS-5, MIT Laboratory for Computer Science, 1989. Lecture notes for 6.852.

[18] N. Lynch and M. Tuttle. An introduction to input/output automata. To be published in *Centrum voor Wiskunde en Informatica Quarterly.* Also in Technical Memo MIT/LCS/TM-373, MIT Laboratory for Computer Science, November 1988.

[19] N. Lynch, Y. Mansour, and A. Fekete. The data link layer: two impossibility results. In *Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing,* pages 149–170, Toronto, Canada, August 1988. Also, Technical Memo MIT/LCS/TM-355, MIT Laboratory for Computer Science, May 1988.

[20] F. Modugno, M. Merritt, and M. Tuttle. Time constrained automata. November 1988. Unpublished manuscript.

[21] M. Tuttle. A game-theoretic characterization of eventual common knowledge. October 1988. Unpublished manuscript.

[22] J. Welch and N. Lynch. Synthesis of efficient drinking philosophers algorithms. In progress.

[23] J. Welch, L. Lamport, and N. Lynch. A lattice-structured proof technique applied to a minimum spanning tree algorithm. In *Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing,* pages 28–43, Toronto, Canada, August 1988. Expanded version in Technical Memo MIT/LCS/TM-361, MIT Laboratory for Computer Science, June 1988.

Theses in Progress

[1] K. Goldman. *Concurrent Algorithm Simulation Using Input/Output Automata.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised by N. Lynch.

[2] J. Leo. *I/O Automata Techniques and Examples.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised by N. Lynch.

[3] S. Ponzio. *A Real Time Analysis of Some Problems in Distributed Computing.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised by N. Lynch.

[4] N. Shavit. *Concurrency in Communication.* PhD thesis, Hebrew University, Jerusalem. Supervised by D. Dolev.

[5] G. Troxel. *A Hierarchical Proof of an Algorithm for Deadlock Recovery in a System using Remote Procedure Calls.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised by N. Lynch.

[6] M. Tuttle. *Knowledge and Distributed Computation.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised N. Lynch.

Theses Completed

[1] C. Colby. *Correctness proofs of the Peterson-Fischer mutual exclusion algorithms.* Bachelor's thesis. MIT Department of Electrical Engineering and Computer Science, June 1989. Supervised by N. Lynch.

[2] M. Nour. *An Automata-theoretic Model for Unity.* Bachelor's thesis. MIT Department of Electrical Engineering and Computer Science, June 1989. Supervised by N. Lynch.

Lectures

H. Attiya. Reliable communication over unreliable channels. Lecture given at IBM T. J. Watson Research Center, December 1988.

H. Attiya. Time bounds for real time process control in the presence of timing uncertainty. Lecture given at IBM Almaden Research Center, April 1989.

N. Lynch. A hundred impossibility proofs for distributed computing. Keynote Address given at the Seventh Annual Symposium on Principles of Distributed Computing. Toronto, Canada, August 1988.

N. Lynch. A theory of atomic transactions. Invited address given at the 2nd International Conference on Database Theory, Bruges, Belgium (August 1988); CWI, Amsterdam (August 1988); Princeton University (July 1989); Harvard University (April 1989); IBM Almaden (April 1989).

N. Lynch. Modeling real time systems. Lecture given at ONR Workshop on Real Time Systems. Fall Church, VA, November 1988.

N. Lynch. Three impossibility results for distributed computing. Lecture given at Digital Equipment Corporation (December 1988); Memorial University, St. Johns, Newfoundland (March 1989); MIT freshman seminar in "schools of thought."

N. Lynch. I/O automata - a model for discrete event systems. Lecture given at LCS Annual Meeting (June 1989); Harvard University (December 1988).

N. Lynch. Multivalued possibilities mappings. Lecture given at REX Workshop on Stepwise Refinement of Distributed Systems, May 1989.

M. Tuttle. Programming simultaneous actions using common knowledge. Lecture given at IBM T.J. Watson Research Center (February), Cornell University (February); DEC Cambridge Research Laboratory (March); Brown University (March); New York University/Courant Institute (March); Princeton University (March); Yale University (April), 1989.

X Consortium

Research Staff

D. Converse	C. Peterson
J. Fulton	R. Scheifler, Group Leader
K. Packard	

Graduate Student

G. Khare

Undergraduate Student

D. Schmidt

Support Staff

M. Leger

15.1 Overview of the MIT X Consortium

The MIT X Consortium was formed in January 1988 to further the development of the X Window System. The major goal of the Consortium is to promote cooperation within the computer industry in the creation of standard software interfaces at all layers in the X Window System environment. MIT's role is to provide the vendor-neutral architectural and administrative leadership required to make this work. The Consortium is financially self-supporting, with membership open to any organization. At present, over 60 companies belong to the Consortium, as well as several universities. These members represent the bulk of the US computer industry, as well as considerable segment of the international industry.

15.2 Current Status and Future Plans

15.2.1 Release 3

One of the primary tasks of the Consortium staff is the maintenance and evolution of a software distribution containing sample implementations of all interfaces defined by the Consortium, as well as numerous applications and utilities. In October, Release 3 of this distribution, consisting of 26 megabytes of source code, was made available to the world, along with a companion collection of roughly 80 megabytes of source code of user-contributed software. The distribution is available using anonymous FTP from a number of Internet sites, and on magnetic tape from the MIT Software Center.

15.2.2 Configuration Management

Configuration management is an important aspect of any large software system, and the X distribution is no exception. Development is performed on more than half a dozen different platforms, each running a different operating system, and the system is used externally on a variety of additional platforms. Details of how to build and install the system vary with every operating system (networking interfaces, program names, compile options, header and library files, etc.) and machine type (low level graphics code is device-specific), as well as with site-specific preferences (where programs should be installed, default values, etc.).

Jim Fulton has done extensive work during the past year on configuration management for the X distribution. Release 3 has been widely praised for its ease of portability and installation, and significant improvements have been made since then.

15.2.3 X Conference

In January, we hosted the Third Annual X Technical Conference. The purpose of the conference is to present and discuss leading edge research and development in the X environment from both academia and industry. This year's conference consisted of seven tutorials, 20 presentations, and 14 informal "birds of a feather" sessions, spread over three days. Donna Converse and Michelle Leger handled the bulk of the organizational details, including registration, scheduling, catering, conference proceedings, and video tape coordination. The conference was attended by approximately 1100 people, free of charge, and was well received. Since attendance has grown every year and we reached the seating capacity of MIT facilities, we expect to hold next year's Conference off-campus.

15.2.4 MIT Research Funding

The X Consortium membership would like to see MIT remain the leader and focal point in research and development of the X Window System. To that end, the X Consortium created a funding pool to encourage MIT faculty, staff, and students to participate in X-related research and development activities. The pool for 1989 was set at $250,000. Although relatively few proposals have been received, two projects have been approved, and approval of several more is anticipated.

15.2.5 Sample Server Implementation

Keith Packard made several improvements to the MIT X server implementation. Using a prototype provided by Adam de Boor at UC Berkeley, Keith wrote a device-independent implementation of backing-store and save-unders (in which the server saves portions of windows that are obscured by other windows so that some exposures can be handled automatically) for Release 3. In addition, Keith spent considerable time and energy producing an implementation for drawing arcs that conforms to the X protocol specification. Although the version that was distributed in Release 3 was rather slow, it has been sped up significantly. In addition, we recently received some very important enhancements from Joel McCormack at Digital Equipment Corporation that optimize region operations and window hierarchy manipulations. Keith added some of his own improvements and simplifications to this code, and integrated it into our implementation.

Up through Release 3, failure to allocate memory would result in server termination. Since catastrophic failure is not a desired response (particularly in limited memory environments such as X terminals), this was of considerable concern. Bob Scheifler has since reworked the server to survive most memory allocation failures, reporting errors back to the requesting client in accordance with the X protocol specification, and continuing to operate normally. The only failures which are not yet handled adequately are those occurring during region operations, principally during window hierarchy reconfiguration. Bob Scheifler and Keith Packard have developed a strategy for gracefully surviving these failures, and future implementation work is planned.

In addition to recovering from allocation failures, overall memory consumption in the server has been significantly reduced. Bob Scheifler and Keith Packard have designed new data structures for the major server resources (windows and graphics contexts) that should cut memory size approximately in half. A further benefit of this work was a revised strategy for layering in the server (based on the usual object-oriented strategy of using wrappers around methods) that will considerably simplify the implementations of backing-store and software cursors.

15.2.6 Standard Colormaps

Standard Colormaps are a mechanism that allows applications to share commonly-used color resources, with an efficient mapping from RGB color values to pixel values for display. Keith Packard and Donna Converse have developed an algorithm for constructing Standard Colormaps which permits other applications (those using the normal X protocol color lookup facilities) to also share the same color resources. This is particularly important on typical color workstations today, which support only a single hardware colormap with a limited number of colormap entries. Keith Packard also developed a revised algorithm for creating Standard Colormaps that are linear ramps through the RGB cube, which now allows applications to treat gray scale and other linear maps to be treated uniformly with all other Standard Colormaps. Donna Converse implemented a set of routines for creating Standard Colormaps for all of the visual classes defined by the X protocol.

15.2.7 Graphics Benchmark

Dan Schmidt, working with Jim Fulton, developed a graphics benchmark and demonstration program. The program provides a simple interface for exercising all of the attributes that can affect graphical output (such as line style, cap style, join style, fill style, dash pattern, and line width) in combination with the various graphical primitives (such as points, lines, arcs, and text). In addition to providing a means for obtaining performance data, this program will also be a valuable interactive demonstration of the X graphics model.

15.2.8 Xt Intrinsics

A major accomplishment in the past year has been standardization (within the Consortium) of the Xt Intrinsics, an object-oriented foundation for building user interface toolkits. Such toolkits (including the MIT Athena Widgets set) are available from a growing number of vendors.

Work on the Intrinsics is far from finished, however. As vendors have begun to use the Intrinsics in earnest, a number of deficiencies (in both function and performance) have been identified. In particular, using a window for every user interface component (called "widgets") caused concern over the amount of memory used in both the client (where the toolkit resides) and in the server. As a result, a proposal for windowless widgets (originally designed and implemented by Digital Equipment Corporation, now used by a number of companies) is currently under review within the Consortium, under the overall guidance of Ralph Swick of MIT Project Athena.

15.2.9 Athena Widgets

Chris Peterson has been doing considerable work fixing and enhancing the Athena Widget set, which is used in a growing number of our core applications. The most significant recent addition is a long-awaited menu widget, supporting both pulldown and popup menus. This widget has now replaced several incompatible menu implementations in our distribution and is expected to be widely used.

15.2.10 Core Components

There are now several product-quality widget sets built on top of the Xt Intrinsics. Although these widget sets all provide remarkably similar functionality, they differ considerably in their graphical user interface (appearance and behavior) and their programmatic interface. For the programmers who wish to have their applications blend in with the other applications on a given vendor's platform, the ability to easily retarget a given application to more than one graphical user interface is crucial. Unfortunately, with present toolkits, the differences in programmatic interfaces makes this task quite difficult in practice.

The Core Components effort is an attempt to specify a policy-free application programmer's interface, that would permit different implementations to embody disparate graphical user interface policies, transparent to the application programmer. Dana Laursen of Hewlett-Packard is the chief architect of the Core Components. Ralph Swick of MIT Project Athena and Bob Scheifler have contributed to the fundamental architecture, and a variety of engineers in several Consortium organizations have contributed to the design. Although considerable progress has been made, a number of very hard problems still exist, such as how to permit subclassing without exposing functionality that is specific to a particular graphical user interface. The time and resource investment required to complete the research now appears to be too large for many Consortium organizations, who must focus in the short term on shipping initial toolkit products.

15.2.11 Inter-client Communication Conventions Manual

The Inter-client Communication Conventions Manual (ICCCM) establishes policies covering a number of mechanisms in the X protocol in order to allow applications from independent vendors to coexist and cooperate in the X environment. The ICCCM covers the use of the selection mechanism for peer-to-peer data exchange (e.g. in cut and paste operations), client to window manager communication (for dealing with title bars, icons, geometry, input focus, colormaps, etc.), client to session manager communication (for client checkpoint and window deletion), and client manipulation of keyboard and pointing device attributes. The overall architect for the ICCCM has been David Rosenthal of Sun Microsystems, with considerable input from engineers in various Consortium organizations and all of the MIT staff. The document is now out for its second public review, and a final standard will be produced shortly Jim Fulton designed and implemented the Xlib changes required to support the ICCCM, and these changes are also now out for public review.

15.2.12 X Display Manager

Keith Packard produced XDM, the X Display Manager, for Release 3. This daemon manages a collection of X displays, including X terminals, on a given host. It provides for authenticating a user at a display (i.e., login) and running the user's session. Although designed to work with both hardware displays local to the host and with remote X terminals, a control protocol is required to make X terminals work well. In particular, when an X terminal is powered on or reset, a mechanism is needed to inform the host's XDM daemon that the terminal is now up, so that login can be initiated. Keith Packard and Bob Scheifler have designed the X Display Manager Control Protocol (XDMCP) for this purpose. The protocol also deals with network security issues, and permits centralized configuration management in an environment with a large number of terminals and potential login hosts. The protocol is currently under review within the Consortium.

15.2.13 Security

Security has long been a low priority issue in the X world. However, with the increase in commercial X products, and with the rash of computer break-ins and viruses over the past year, interest in security is now rather high. The default host-based access control mechanism in the core X protocol is simply not adequate in most environments. Keith Packard and Jim Fulton have designed and implemented a basic framework for allowing X clients to send authorization information to the X server, and Keith Packard put a simple encryption-based authorization scheme into the X Display Manager and the X server as a test. A more secure scheme is being worked on as part of the X Display Manager Control Protocol, and work is ongoing at MIT Project Athena to integrate Kerberos as an authorization mechanism.

15.2.14 Compound Text

Internationalization (or localization) of user interface software is increasingly important to X vendors. A key aspect of this is dealing with text in languages other than English. There are three important uses of text in the X environment that are external to applications: inter-client communication using selections (e.g. cut and paste); window properties (e.g. text for title bars); and resources (e.g. text for labels and prompts). Typically, different languages have different character sets, and each character set is given a particular encoding (usually one or two bytes per character). In some cases, the characters used for a single language are split across more than one character set encoding.

Bob Scheifler developed a format for multiple character set data, called Compound Text, based on ISO standards for encoding and combining character sets (ISO 2022 and ISO 6429). Compound Text is intended to be an external representation, or interchange format, for use in the three areas listed above. It is not intended to be an internal representation within an application; it is expected (but not required) that clients will convert Compound Text to some internal representation for processing and rendering, and convert from that internal representation to Compound String when providing textual data to another client. The format supports the standard ISO 8859 character set encodings and the standard Japanese, Chinese, and Korean character set encodings, and encourages their use, but also allows non-standard encodings to be used. Horizontal direction of text can also be encoded. The Compound Text specification is now out for public review.

15.2.15 X Logical Font Description Conventions

Jim Flowers of Digital Equipment has been the chief architect of the X Logical Font Description Conventions, which establish a standard parsable font name format and standard font properties, providing X clients a server-independent means to query and use a rich collection of fonts. For example, the conventions provide an adequate set of typographic font attributes for publishing and other applications to do intelligent font matching or substitution when handling documents; automatically place subscripts and superscripts; and determine small capital heights, recommended leading, and wordspace values. Bob Scheifler contributed to the design, and Jim Fulton worked to ensure that fonts contributed to MIT and font support mechanisms (such as the font compiler) conform to the conventions. The conventions are now out for public review prior to finalizing the standard.

15.2.16 Font Server

With the advent of X terminals and other limited-memory servers, and the significant increase in quality screen fonts, the idea of a font server is now attractive. A font server is a program for providing font information to client programs (such as X servers, or even print servers and document previewers). Like the X server, the font server will be able to simultaneously support multiple clients of differing architectures over any virtual stream connection. The font server should be able to deal with multiple font (input and output) formats, and provide partial font information (so that limited client memory can be used as a cache). The font server must also be able to enforce licensing restrictions on a per-font basis. Tom Porcher of Digital Equipment formulated a requirements document for font service, and Jim Fulton put together a preliminary design for a font server protocol. Daniel Dardailler of Bull is interested in pursuing the design and implementation to completion, and we hope for more progress next year.

15.2.17 PEX Sample Implementation

PEX is a 3D graphics extension to the X protocol, supporting the PHIGS and PHIGS+ graphics interfaces. In order to establish proof of concept of the PEX design, and to promote the use of PEX, an effort is now underway to build a sample implementation of both the PEX server extension and a full client library (providing the PHIGS and PHIGS+ programming interface). Bob Scheifler, working with several interested companies, put together a Request

For Proposals. A number of bids were received, and one was selected based on input from potential sponsors. There are now 15 sponsors of the implementation which is scheduled for release to the public in the early spring of 1991.

15.2.18 Input Device Extension

The core X protocol deals only with one keyboard and one pointing device. However, many workstations have an assortment of input devices which would be very useful to X applications. George Sachs of Hewlett-Packard and Mark Patrick of Ardent Computer produced an X protocol extension and a C library for dealing with additional input devices. The primary devices supported are those with keys, buttons, and one or more axes of motion. In addition, the extension is designed to itself be extensible, so that new classes of input devices, and new combinations of classes, can easily be added. Bob Scheifler and Keith Packard contributed to the design, as have engineers from a number of Consortium organizations.

15.2.19 Video Extension

Todd Brunhoff of Tektronix has been working on an X protocol extension to provide an X interface to the generally interesting aspects of displaying live video in windows, capturing graphics from windows and converting them to a video signal, and managing the network of connections to and from devices that may receive or produce these signals, such as video tape recorders.

15.2.20 Multi-buffering and Stereo Extension

Jeff Friedberg and Larry Seiler of Digital Equipment, and Jeff Vroom of Stellar Computer, worked to merge several different double-buffering proposals into a single coherent X protocol extension for supporting multi-buffering and stereoscopic viewing of windows. The extension allows multiple, independently-addressable output buffers to be associated with normal and stereo windows. Any of the buffers can be displayed in the window, and a series of buffers can be displayed in rapid succession to achieve a smooth animation. Keith Packard and Bob Scheifler contributed to the design, which is now out for public review.

15.2.21 Nonrectangular Windows

Keith Packard designed and implemented an X protocol extension for changing the visible shape of a window to an arbitrary nonrectangular shape, including forms composed of disjoint pieces. Each window is defined by two regions: the bounding region and the clip region. The bounding region is the area of the parent window which the window will occupy (including border). The clip region is the subset of the bounding region which is available for subwindows and graphics. The area between the bounding region and the clip region is defined to be the border of the window. The extension proved remarkably simple to implement in the server, with changes required in just a few places. The added functionality imposes no cost penalty on rectangular windows, and performance for nonrectangular windows is quite acceptable. The extension is now under review within the Consortium.

15.2.22 X Testing Consortium

The X Testing Consortium is a loosely bound group of approximately a dozen companies working together on comprehensive test software for the X protocol and the Xlib C language interface to it. Bob Scheifler has been meeting with this group over the past year, providing some guidance. The output of the group will be both informal test specifications, and code implementing those specifications. The group also did some work on performance benchmarks, and Jim Fulton provided review and feedback on that work. Producing a complete test suite proved to require considerably more effort than expected, and various companies are now pulling resources off the project to work on other tasks. The group will be producing a final but incomplete release soon. Bob Scheifler is working on plans to continue the effort within the X Consortium, and to expand the testing effort to cover other components of the X Window System.

15.2.23 Formal Standards

The X3H3.6 Window System Task Group, under the X3H3 Computer Graphics Standards Committee, under ANSI, has been working on formal standardization of the X protocol for about two years now. Bob Scheifler has been attending the X3H3.6 meetings and participating in their deliberations. The protocol specification has recently gone out for letter ballot within X3H3, but an ANSI standard is still perhaps fourteen months away. The task group had planned to start work on Xlib by now, but lack of resources have made that impractical. Both the task group and the X Consortium have recently approved a working relationship, in which X3H3.6 will ask the X Consortium to develop resolutions to technical issues as they arise during the remainder of the ANSI process.

The IEEE Technical Committee on Operating Systems formed a new working group, P1201.1, to formally standardize (under ANSI) toolkit functionality and behavior in the X environment. Bob Scheifler and Chris Peterson have to date shared responsibilities for interacting with this group. Their immediate goal appears to be standardization of a widget set based on the Xt Intrinsics, but to do so requires that the Intrinsics and Xlib be standardized. The group does not currently have the resources to do this, and is coordinating with X3H3.6 in an attempt to develop a workable plan.

The National Institute of Standards and Technology issued a proposed Federal Information Processing Standard for the X Window System, composed of the standard specifications in Release 3 of the X Consortium software distribution: the X protocol, the Xlib C binding, the Xt Intrinsics, and the Bitmap Distribution Format for fonts. The fact that NIST issued this

FIPS without waiting for formal standardization by ANSI caused considerable consternation in the formal standards world, and the fact that the FIPS appears to make the Intrinsics an exclusive federal standard (the Intrinsics are a non-exclusive standard in the X Consortium) caused considerable consternation on the part of certain X vendors. Bob Scheifler has been working with all of these players to ensure that technical arguments are correct, and to try to keep the political arguments in perspective.

15.2.24 Registration

We established a mechanism to allow the X community to register the following significant items: organization names (used as prefixes for other names), keysyms, authorization protocol names, vendor server string formats, protocol extension names, host address family formats, window property names and types, selection names and targets, window manager protocols, font foundry names, font property names, resource types, and application classes. The primary goal of registration is to avoid conflicting use of a given name or value. The secondary goal is to encourage use of these items by more than one organization.

15.2.25 Graphical Programming Environment for Configuration

Geeta Khare explored the specification, design, and implementation of a graphical programming language to address a class of problems known as configuration problems, those concerned with the correctness and completeness of a collection of items and/or the arrangement of those items under a set of constraints. Domain specificity of the language is achieved through the types and operations provided. A configuration task grammar was produced, which lists the breakdown of configuration tasks into subtasks and primitives, and a categorization was produced of objects found in configuration problems, allowing reuse of primitives in different configuration problems. A prototype was implemented in Common Lisp using KnowledgeCraft.

15.3 Publications

[1] T. Brunhoff, Tektronix. *VEX: Video Extension to X.*

[2] J. Flowers, Digital Equipment Corporation. *X Logical Font Description Conventions.*

[3] J. Friedberg, Digital Equipment Corporation; L. Seiler, Digital Equipment Corporation; and J. Vroom, Stellar Computer. *Extending X for Double-Buffering, Multi-Buffering, and Stereo.*

[4] J. Fulton. *Configuration Management on the X Window System.* Usenix Association, Berkeley CA, April 1989.

[5] D. Laursen, Hewlett-Packard Company. *Core Component Widgets.*

[6] J. McCormack, Digital Equipment Corporation; P. Asente, Digital Equipment Corporation; and Ralph Swick, MIT Project Athena. *X Toolkit Intrinsics: C Language Interface.*

[7] K. Packard. *X Display Manager Control Protocol.* MIT X Consortium.

[8] K. Packard. *X11 Nonrectangular Window Shape Extension.* MIT X Consortium.

[9] M. Patrick, Ardent Computer and G. Sachs, Hewlett-Packard Company. *X11 Input Extension Protocol Specification.*

[10] M. Patrick, Ardent Computer and G. Sachs, Hewlett-Packard Company. *X11 Input Extension Library Specification.*

[11] D. Rosenthal. *Inter-Client Communication Conventions Manual.* Sun Microsystems.

[12] R. Rost, Digital Equipment Corporation. *PEX Introduction and Overview.*

[13] R. Rost, Digital Equipment Corporation. *PEX Protocol Specification.*

[14] R. Scheifler, J. Gettys, and R. Newman, *X Window System C Library and Protocol Reference.* Digital Press, Bedford, MA, 1988.

[15] R. Scheifler. *Compound Text Encoding.* MIT X Consortium.

[16] R. Swick, MIT Project Athena and T. Weissman, Digital Equipment Corporation. *X Toolkit Athena Widgets: C Language Interface.*

[17] Third Annual X Technical Conference, January 1989, video tapes.

Thesis Completed

[1] G. Khare. *A Graphical Programming Environment for Configuration.* Master's thesis, MIT, June 1989.

Publications

Technical Memos[1]

TM-10 Jackson, J.N.
Interactive Design Coordination for the Building Industry, June 1970; AD 708400

TM-11 Ward, P.W.
Description and Flow Chart of the PDP-7/9 Communications Package, July 1970; AD 711379

TM-12 Graham, R.M.
File Management and Related Topics, June 12, 1970, September 1970; AD 712068

TM-13 Graham, R.M.
Use of High Level Languages for Systems Programming, September 1970; AD 711965

TM-14 Vogt, C.M.
Suspension of Processes in a Multi-processing Computer System, September 1970; AD 713989

TM-15 Zilles, S.N.
An Expansion of the Data Structuring Capabilities of PAL, October 1970; AD 720761

TM-16 Bruere-Dawson, G.
Pseudo-Random Sequences, October 1970; AD 713852

TM-17 Goodman, L.I.
Complexity Measures for Programming Languages, September 1971; AD 729011

TM-18 Reprinted as TR-85

TM-19 Fenichel, R.R.
A New List-Tracing Algorithm, October 1970; AD 714522

TM-20 Jones, T.L.
A Computer Model of Simple Forms of Learning, January 1971; AD 720337

TM-21 Goldstein, R.C.
The Substantive Use of Computers For Intellectual Activities, April 1971; AD 721618

TM-22 Wells, D.M.
Transmission of Information Between A Man-Machine Decision System and its Environment, April 1971; AD 722837

TM-23 Strnad, A.J.
The Relational Approach to the Management of Data Bases, April 1971; AD 721619

TM-24 Goldstein, R.C. and Strnad, A.J.
The MacAIMS Data Management System, April 1971; AD 721620

[1]TMs 1-9 were never issued.

TM-25 Goldstein, R.C.
Helping People Think, April 1971; AD 721998

TM-26 Iazeolla, G.G.
Modeling and Decomposition of Information Systems for Performance Evaluation, June 1971; AD 733965

TM-27 Bagchi, A.
Economy of Descriptions and Minimal Indices, January 1972, AD 736960

TM-28 Wong, R.
Construction Heuristics for Geometry and a Vector Algebra Representation of Geometry, June 1972, AD 743487

TM-29 Hossley, R. and Rackoff, C.
The Emptiness Problem for Automata on Infinite Trees, June 1972; AD 747250

TM-30 McCray, W.M.
SIM360: A S/360 Simulator, October 1972; AD 749365

TM-31 Bonneau, R.J.
A Class of Finite Computation Structures Supporting the Fast Fourier Transform, March 1973; AD 757787

TM-32 Moll, R.
An Operator Embedding Theorem for Complexity Classes of Recursive Functions, May 1973; AD 759999

TM-33 Ferrante, J. and Rackoff, C.
A Decision Procedure for the First Order Theory of Real Addition with Order, May 1973; AD 760000

TM-34 Bonneau, R.J.
Polynomial Exponentiation: The Fast Fourier Transform Revisited, June 1973; PB 221-742

TM-35 Bonneau, R.J.
An Interactive Implementation of the Todd-Coxeter Algorithm, December 1973; AD 770565

TM-36 Geiger, S.P.
A User's Guide to the Macro Control Language, December 1973; AD 771435

TM-37 Schonhage, A.
Real-Time Simulation of Multidimensional Turing Machines by Storage Modification Machines, December 1973; PB 226-103/AS

TM-38 Meyer, A.R.
Weak Monadic Second Order Theory of Successor Is Not Elementary-Recursive, December 1973; PB 226-514/AS

TM-39 Meyer, A.R.
Discrete Computation: Theory and Open Problems, January 1974; PB 226-836/AS

TM-40 Paterson, M.S., Fischer, M.J. and Meyer, A.R.
An Improved Overlap Argument for On-Line Multiplication, January 1974; AD 773137

TM-41 Fischer, M.J. and Paterson, M.S.
String-Matching and Other Products, January 1974; AD 773138

TM-42 Rackoff, C.
On the Complexity of the Theories of Weak Direct Products, January 1974; PB 228-459/AS

TM-43 Fischer, M.J. and Rabin, M.O.
Super-Exponential Complexity of Presburger Arithmetic, February 1974; AD 775004

TM-44 Pless, V.
Symmetry Codes and their Invariant Subcodes, May 1974; AD 780243

TM-45 Fischer, M.J. and Stockmeyer, L.J.
Fast On-Line Integer Multiplication, May 1974; AD 779889

TM-46 Kedem, Z.M.
Combining Dimensionality and Rate of Growth Arguments for Establishing Lower Bounds on the Number of Multiplications, June 1974; PB 232-969/AS

TM-47 Pless, V.
Mathematical Foundations of Flip-Flops, June 1974; AD 780901

TM-48 Kedem, Z.M.
The Reduction Method for Establishing Lower Bounds on the Number of Additions, June 1974; PB 233-538/AS

TM-49 Pless, V.
Complete Classification of (24,12) and (22,11) Self-Dual Codes, June 1974; AD 781335

TM-50 Benedict, G.G.
An Enciphering Module for Multics, S.B. Thesis, EE Department, July 1974; AD 782658

TM-51 Aiello, J.M.
An Investigation of Current Language Support for the Data Requirements of Structured Programming, S.M. & E.E. Thesis, EE Department, September 1974; PB 236-815/AS

TM-52 Lind, J.C.
Computing in Logarithmic Space, September 1974; PB 236-167/AS

TM-53 Bengelloun, S.A.
MDC-Programmer: A Muddle-to-Datalanguage Translator for Information Retrieval, S.B. Thesis, EE Department, October 1974; AD 786754

TM-54 Meyer, A.R.
The Inherent Computation Complexity of Theories of Ordered Sets: A Brief Survey, October 1974; PB 237-200/AS

TM-55 Hsieh, W.N., Harper, L.H. and Savage, J.E.
A Class of Boolean Functions with Linear Combinatorial Complexity, October 1974; PB 237-206/AS

TM-56 Gorry, G.A.
Research on Expert Systems, December 1974

TM-57 Levin, M.
On Bateson's Logical Levels of Learning, February 1975

TM-58 Qualitz, J.E.
Decidability of Equivalence for a Class of Data Flow Schemas, March 1975; PB 237-033/AS

TM-59 Hack, M.
Decision Problems for Petri Nets and Vector Addition Systems, March 1975; PB 231-916/AS

TM-60 Weiss, R.B.
CAMAC: Group Manipulation System, March 1975; PB 240-495/AS

TM-61 Dennis, J.B.
First Version of a Data Flow Procedure Language, May 1975

TM-62 Patil, S.S.
An Asynchronous Logic Array, May 1975

TM-63 Pless, V.
Encryption Schemes for Computer Confidentiality, May 1975; A010-217

TM-64 Weiss, R.B.
Finding Isomorph Classes for Combinatorial Structures, S.M. Thesis, EE Department, June 1975

TM-65 Fischer, M.J.
The Complexity Negation-Limited Networks - A Brief Survey, June 1975

TM-66 Leung, C.K.C.
Formal Properties of Well-Formed Data, June 1975.

TM-67 Cardoza, E.E.
Computational Complexity of the World Problem for Commutative Semigroups, S.M. Thesis, EE & CS Department, October 1975

TM-68 Weng, K-S.
Stream-Oriented Computation in Recursive Data Flow Schemas, S.M. Thesis, EE & CS Department, October 1975

TM-69 Bayer, P.J.
Improved Bounds on the Costs of Optimal and Balanced Binary Search Trees, S.M. Thesis, EE & CS Department, November 1975

TM-70 Ruth, G.R.
Automatic Design of Data Processing Systems, February 1976; A023-451

TM-71 Rivest, R.
On the Worst-Case of Behavior of String-Searching Algorithms, April 1976

TM-72 Ruth, G.R.
Protosystem I: An Automatic Programming System Prototype, July 1976; A026-912

TM-73 Rivest, R.
Optimal Arrangement of Keys in a Hash Table, July 1976

TM-74 Malvania, N.
The Design of a Modular Laboratory for Control Robotics, S.M. Thesis, EE & CS Department, September 1976; A030-418

TM-75 Yao, A.C. and Rivest, R.
K+1 Heads are Better than K, September 1976; A030-008

TM-76 Bloniarz, P.A., Fischer, M.J. and Meyer, A.R.
A Note on the Average Time to Compute Transitive Closures, September 1976

TM-77 Mok, A.K.
Task Scheduling in the Control Robotics Environment, S.M. Thesis, EE & CS Department, September 1976; A030-402

TM-78 Benjamin, A.J.
Improving Information Storage Reliability Using a Data Network, S.M. Thesis, EE & CS Department, October 1976; A033-394

TM-79 Brown, G.P.
A System to Process Dialogue: A Progress Report, October 1976; A033-276

TM-80 Even, S.
The Max Flow Algorithm of Dinic and Karzanov: An Exposition, December 1976

TM-81 Gifford, D.
Hardware Estimation of a Process' Primary Memory Requirements, S.B. Thesis, EE & CS Department, January 1977

TM-82 Rivest, R.L., Shamir, A. and Adelman, L.M.
A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, April 1977; A039-036

TM-83 Baratz, A.E.
Construction and Analysis of Network Flow Problem which Forces Karzanov Algorithm to $O(n^3)$ Running Time, April 1977

TM-84 Rivest, R.L. and Pratt, V.R.
The Mutual Exclusion Problem for Unreliable Processes, April 1977

TM-85 Shamir, A.
Finding Minimum Cutsets in Reducible Graphs, June 1977; A040-698

TM-86 Szolovits, P., Hawkinson, L.B. and Martin, W.A.
An Overview of OWL, A Language for Knowledge Representation, June 1977; A041-372

TM-87 Clark, D., Editor
Ancillary Reports: Kernel Design Project, June 1977

TM-88 Lloyd, E.L.
On Triangulations of a Set of Points in the Plane, S.M. Thesis, EE & CS Department, July 1977

TM-89 Rodriguez, H.
Measuring User Characteristics on the Multics System, S.B. Thesis, EE & CS Department, August 1977

TM-90 d'Oliveira, C.R.
An Analysis of Computer Decentralization, S.B. Thesis, EE & CS Department, October 1977; A045-526

TM-91 Shamir, A.
Factoring Numbers in $O(\log n)$ Arithmetic Steps, November 1977; A047-709

TM-92 Misunas, D.P.
Report on the Workshop on Data Flow Computer and Program Organization, November 1977

TM-93 Amikura, K.
A Logic Design for the Cell Block of a Data-Flow Processor, S.M. Thesis, EE & CS Department, December 1977

TM-94 Berez, J.M.
A Dynamic Debugging System for MDL, S.B. Thesis, EE & CS Department, January 1978; A050-191

TM-95 Harel, D.
Characterizing Second Order Logic With First Order Quantifiers, March 1977

TM-96 Harel, D., Pnueli, A. and Stavi, J.
A Complete Axiomatic System for Proving Deductions about Recursive Programs, February 1978

TM-97 Harel, D., Meyer, A.R. and Pratt, V.R.
Computability and Completeness in Logics of Programs, February 1978

TM-98 Harel, D. and Pratt, V.R.
Nondeterminism in Logics of Programs, February 1978

TM-99 LaPaugh, A.S.
The Subgraph Homeomorphism Problem, S.M. Thesis, EE & CS Department, February 1978

TM-100 Misunas, D.P.
A Computer Architecture for Data-Flow Computation, S.M. Thesis, EE & CS Department, March 1978; A052-538

TM-101 Martin, W.A.
Descriptions and the Specialization of Concepts, March 1978; A052-773

TM-102 Abelson, H.
Lower Bounds on Information Transfer in Distributed Computations, April 1978

TM-103 Harel, D.
Arithmetical Completeness in Logics of Programs, April 1978

TM-104 Jaffe, J.
The Use of Queues in the Parallel Data Flow Evaluation of "If-Then-While" Programs, May 1978

TM-105 Masek, W.J. and Paterson, M.S.
A Faster Algorithm Computing String Edit Distances, May 1978

TM-106 Parikh, R.
A Completeness Result for a Propositional Dynamic Logic, July 1978

TM-107 Shamir, A.
A Fast Signature Scheme, July 1978; A057-152

TM-108 Baratz, A.E.
An Analysis of the Solovay and Strassen Test for Primality, July 1978

TM-109 Parikh, R.
Effectiveness, July 1978

TM-110 Jaffe, J.M.
An Analysis of Preemptive Multiprocessor Job Scheduling, September 1978

TM-111 Jaffe, J.M.
Bounds on the Scheduling of Typed Task Systems, September 1978

TM-112 Parikh, R.
A Decidability Result for a Second Order Process Logic, September 1978

TM-113 Pratt, V.R.
A Near-optimal Method for Reasoning about Action, September 1978

TM-114 Dennis, J.B., Fuller, S.H., Ackerman, W.B., Swan, R.J. and Weng, K-S.
Research Directions in Computer Architecture, September 1978; A061-222

TM-115 Bryant, R.E. and Dennis, J.B.
Concurrent Programming, October 1978; A061-180

TM-116 Pratt, V.R.
Applications of Modal Logic to Programming, December 1978

TM-117 Pratt, V.R.
Six Lectures on Dynamic Logic, December 1978

TM-118 Borkin, S.A.
Data Model Equivalence, December 1978; A062-753

TM-119 Shamir, A. and Zippel, R.E.
On the Security of the Merkle-Hellman Cryptographic Scheme, December 1978; A063-104

TM-120 Brock, J.D.
Operational Semantics of a Data Flow Language, S.M. Thesis, EE & CS Department, December 1978; A062-997

TM-121 Jaffe, J.
The Equivalence of R.E. Programs and Data Flow Schemes, January 1979

TM-122 Jaffe, J.
Efficient Scheduling of Tasks Without Full Use of Processor Resources, January 1979

TM-123 Perry, H.M.
An Improved Proof of the Rabin-Hartmanis-Stearns Conjecture, S.M. & E.E. Thesis, EE & CS Department, January 1979

TM-124 Toffoli, T.
Bicontinuous Extensions of Invertible Combinatorial Functions, January 1979; A063-886

TM-125 Shamir, A., Rivest, R.L. and Adelman, L.M.
Mental Poker, February 1979; A066-331

TM-126 Meyer, A.R. and Paterson, M.S.
With What Frequency Are Apparently Intractable Problems Difficult?, February 1979

TM-127 Strazdas, R.J.
A Network Traffic Generator for DECNET, S.B. & S.M. Thesis, EE & CS Department, March 1979

TM-128 Loui, M.C.
Minimum Register Allocation is Complete in Polynomial Space, March 1979

TM-129 Shamir, A.
On the Cryptocomplexity of Knapsack Systems, April 1979; A067-972

TM-130 Greif, I. and Meyer, A.R.
Specifying the Semantics of While-Programs: A Tutorial and Critique of a Paper by Hoare and Lauer, April 1979; A068-967

TM-131 Adelman, L.M.
Time, Space and Randomness, April 1979

TM-132 Patil, R.S.
Design of a Program for Expert Diagnosis of Acid Base and Electrolyte Disturbances, May 1979

TM-133 Loui, M.C.
The Space Complexity of Two Pebble Games on Trees, May 1979

TM-134 Shamir, A.
How to Share a Secret, May 1979; A069-397

TM-135 Wyleczuk, R.H.
Timestamps and Capability-Based Protection in a Distributed Computer Facility, S.B. & S.M. Thesis, EE & CS Department, June 1979

TM-136 Misunas, D.P.
Report on the Second Workshop on Data Flow Computer and Program Organization, June 1979

TM-137 Davis, E. and Jaffe, J.M.
Algorithms for Scheduling Tasks on Unrelated Processors, June 1979

TM-138 Pratt, V.R.
Dynamic Algebras: Examples, Constructions, Applications, July 1979

TM-139 Martin, W.A.
Roles, Co-Descriptors, and the Formal Representation of Quantified English Expressions, September 1979; A074-625

TM-140 Szolovits, P.
Artificial Intelligence and Clinical Problem Solving, September 1979

TM-141 Hammer, M.M. and McLeod, D.
On Data Base Management System Architecture, October 1979; A076-417

TM-142 Lipski, W.
On Data Bases with Incomplete Information, October 1979

TM-143 Leth, J.W.
An Intermediate Form for Data Flow Programs, S.M. Thesis, EE & CS Department, November 1979

TM-144 Takagi, A.
Concurrent and Reliable Updates of Distributed Databases, November 1979

TM-145 Loui, M.C.
A Space Bound for One-Tape Multidimensional Turing Machines, November 1979

TM-146 Aoki, D.J.
A Machine Language Instruction Set for a Data Flow Processor, S.M. Thesis, EE & CS Department, December 1979

TM-147 Schroeppel, R. and Shamir, A.
$AT + 0(2^{n/2}), S = 0(2^{n/4})$ Algorithm for Certain NP-Complete Problems, January 1980; A080-385

TM-148 Adelman, L.M. and Loui, M.C.
Space-Bounded Simulation of Multitape Turing Machines, January 1980

TM-149 Pallottino, S. and Toffoli, T.
An Efficient Algorithm for Determining the Length of the Longest Dead Path in an "Lifo" Branchand-Bound Exploration Schema, January 1980; A079-912

TM-150 Meyer, A.R.
Ten Thousand and One Logics of Programming, February 1980

TM-151 Toffoli, T.
Reversible Computing, February 1980; A082-021

TM-152 Papadimitriou, C.H.
On the Complexity of Integer Programming, February 1980

TM-153 Papadimitriou, C.H.
Worst-Case and Probabilistic Analysis of a Geometric Location Problem, February 1980

TM-154 Karp, R.M. and Papadimitriou, C.H.
On Linear Characterizations of Combinatorial Optimization Problems, February 1980

TM-155 Itai, A., Lipton, R.J., Papadimitriou, C.H. and Rodeh, M.
Covering Graphs by Simple Circuits, February 1980

TM-156 Meyer, A.R. and Parikh, R.
Definability in Dynamic Logic, February 1980

TM-157 Meyer, A.R. and Winklmann, K.
On the Expressive Power of Dynamic Logic, February 1980

TM-158 Stark, E.W.
Semaphore Primitives and Starvation-Free Mutual Exclusion, S.M. Thesis, EE & CS Department, March 1980

TM-159 Pratt, V.R.
Dynamic Algebras and the Nature of Induction, March 1980

TM-160 Kanellakis, P.C.
On the Computational Complexity of Cardinality Constraints in Relational Databases March 1980

TM-161 Lloyd, E.L.
Critical Path Scheduling of Task Systems with Resource and Processor Constraints, March 1980

TM-162 Marcum, A.M.
A Manager for Named, Permanent Objects, S.B. & S.M. Thesis, EE & CS Department, April 1980; A083-491

TM-163 Meyer, A.R. and Halpern, J.Y.
Axiomatic Definitions of Programming Languages: A Theoretical Assessment, April 1980

TM-164 Shamir, A.
The Cryptographic Security of Compact Knapsacks—Preliminary Report, April 1980; A084-456

TM-165 Finseth, C.A.
Theory and Practice of Text Editors or A Cookbook for an Emacs, S.B. Thesis, EE & CS Department, May 1980

TM-166 Bryant, R.E.
Report on the Workshop on Self-Timed Systems, May 1980

TM-167 Pavelle, R. and Wester, M.
Computer Programs for Research in Gravitation and Differential Geometry, June 1980

TM-168 Greif, I.
Programs for Distributed Computing: The Calendar Application, July 1980; A087-357

TM-169 Burke, G. and Moon, D.
LOOP Iteration Macro, July 1980; A087-372

TM-170 Ehrenfeucht, A., Parikh, R. and Rozenberg, G.
Pumping Lemmas for Regular Sets, August 1980

TM-171 Meyer, A.R.
What is a Model of the Lambda Calculus?, August 1980

TM-172 Paseman, W.G.
Some New Methods of Music Synthesis, S.M. Thesis, EE & CS Department, August 1980; A090-130

TM-173 Hawkinson, L.B.
XLMS: A Linguistic Memory System, September 1980; A090-033

TM-174 Arvind, Kathail, V. and Pingali, K.
A Dataflow Architecture with Tagged Tokens, September 1980

TM-175 Meyer, A.R., Weise, D. and Loui, M.C.
On Time Versus Space III, September 1980

TM-176 Seaquist, C.R.
A Semantics of Synchronization, S.M. Thesis, EE & CS Department, September 1980; A091-015

TM-177 Sinha, M.K.
TIMEPAD - A Performance Improving Synchronization Mechanism for Distributed Systems, September 1980

TM-178 Arvind and Thomas, R.E.
I-Structures: An Efficient Data Type for Functional Languages, September 1980

TM-179 Halpern, J.Y. and Meyer, A.R.
Axiomatic Definitions of Programming Languages, II, October 1980

TM-180 Papadimitriou, C.H.
A Theorem in Data Base Concurrency Control, October 1980

TM-181 Lipski, W. and Papadimitriou, C.H.
A Fast Algorithm for Testing for Safety and Detecting Deadlocks in Locked Transaction Systems, October 1980

TM-182 Itai, A., Papadimitriou, C.H. and Szwarefiter, J.L.
Hamilton Paths in Grid Graphs, October 1980

TM-183 Meyer, A.R.
A Note on the Length of Craig's Interpolants, October 1980

TM-184 Lieberman, H. and Hewitt, C.
A Real Time Garbage Collector that can Recover Temporary Storage Quickly, October 1980

TM-185 Kung, H-T. and Papadimitriou, C.H.
An Optimality Theory of Concurrency Control for Databases, November 1980; A092-625

TM-186 Szolovits, P. and Martin, W.A.
BRAND X Manual, November 1980; A093-041

TM-187 Fischer, M.J., Meyer, A.R. and Paterson, M.S.
$W(nlogn)$ Lower Bounds on Length of Boolean Formulas, November 1980

TM-188 Mayr, E.W.
An Effective Representation of the Reachability Set of Persistent Petri Nets, January 1981

TM-189 Mayr, E.W.
Persistence of Vector Replacement Systems is Decidable, January 1981

TM-190 Ben-Ari, M., Halpern, J.Y. and Pnueli, A.
Deterministic Propositional Dynamic Logic: Finite Models, Complexity, and Completeness, January 1981

TM-191 Parikh, R.
Propositional Dynamic Logics of Programs: A Survey, January 1981

TM-192 Meyer, A.R., Streett, R.S. and Mirkowska, G.
The Deducibility Problem in Propositional Dynamic Logic, February 1981

TM-193 Yannakakis, M. and Papadimitriou, C.H.
Algebraic Dependencies, February 1981

TM-194 Barendregt, H. and Longo, G.
Recursion Theoretic Operators and Morphisms on Numbered Sets, February 1981

TM-195 Barber, G.R.
Record of the Workshop on Research in Office Semantics, February 1981

TM-196 Bhatt, S.N.
On Concentration and Connection Networks, S.M. Thesis, EE & CS Department, March 1981

TM-197 Fredkin, E. and Toffoli, T.
Conservative Logic, May 1981

TM-198 Halpern, J.Y. and Reif, J.H.
The Propositional Dynamic Logic of Deterministic, Well-Structured Programs, March 1981

TM-199 Mayr, E.W. and Meyer, A.R.
The Comple y of the Word Problems for Commutative Semigroups and Polynomial Ideals, June 1981

TM-200 Burke, G.S.
LSB Manual, June 1981

TM-201 Meyer, A.R.
What is a Model of the Lambda Calculus? Expanded Version, July 1981

TM-202 Saltzer, J.H.
Communication Ring Initialization without Central Control, December 1981

TM-203 Bawden, A., Burke, G. and Hoffman, C.W.
MACLISP Extensions, July 1981

TM-204 Halpern, J.Y.
On the Expressive Power of Dynamic Logic, II, August 1981

TM-205 Kannon, R.
Circuit-Size Lower Bounds and Non-Reducibility to Sparce Sets, October 1981

TM-206 Leiserson, C.E. and Pinter, R.Y.
Optimal Placement for River Routing, October 1981

TM-207 Longo, G.
Power Set Models For Lambda-Calculus: Theories, Expansions, Isomorphisms, November 1981

TM-208 Cosmadakis, S. and Papadimitriou, C.H.
The Traveling Salesman Problem with Many Visits to Few Cities, November 1981

TM-209 Johnson, D. and Papadimitriou, C.H.
Computational Complexity and the Traveling Salesman Problem, December 1981

TM-210 Greif, I.
Software for the 'Roles' People Play, February 1982

TM-211 Meyer, A.R. and Tiuryn, J.
A Note on Equivalences Among Logics of Programs, December 1981

TM-212 Elias, P.
Minimax Optimal Universal Codeword Sets, January 1982

TM-213 Greif, I.
PCAL: A Personal Calendar, January 1982

TM-214 Meyer, A.R. and Mitchell, J.C.
Terminations for Recursive Programs: Completeness and Axiomatic Definability, March 1982

TM-215 Leiserson, C.E. and Saxe, J.B.
Optimizing Synchronous Systems, March 1982

TM-216 Church, K.W. and Patil, R.S.
Coping with Syntactic Ambiguity or How to Put the Block in the Box on the Table, April 1982

TM-217 Wright, K.D.
A File Transfer Program for a Personal Computer, April 1982

TM-218 Greif, I.
Cooperative Office Work, Teleconferencing and Calendar Management: A Collection of Papers, May 1982

TM-219 Jouannaud, J-P., Lescanne, P. and Reinig, F.
Recursive Decomposition Ordering and Multiset Orderings, June 1982

TM-220 Chu, T-A.
Circuit Analysis of Self-Timed Elements for NMOS VLSI Systems, May 1982

TM-221 Leighton, F.T., Lepley, M. and Miller, G.L.
Layouts for the Shuffle-Exchange Graph Based on the Complex Plane Diagram, June 1982

TM-222 Meier zu Sieker, F.
A Telex Gateway for the Internet, S.B. Thesis, Electrical Engineering Department, May 1982

TM-223 diSessa, A.A.
A Principled Design for an Integrated Computation Environment, July 1982

TM-224 Barber, G.
Supporting Organizational Problem Solving with a Workstation, July 1982

TM-225 Barber, G. and Hewitt, C.
Foundations for Office Semantics, July 1982

TM-226 Bergstra, J., Chmielinska, A. and Tiuryn, J.
Hoares' Logic Not Complete When it Could Be, August 1982

TM-227 Leighton, F.T.
New Lower Bound Techniques for VLSI, August 1982

TM-228 Papadimitriou, C.H. and Zachos, S.K.
Two Remarks on the Power of Counting, August 1982

TM-229 Cosmadakis, S.S.
The Complexity of Evaluation Relational Queries, August 1982

TM-230 Shamir, A.
Embedding Cryptographic Trapdoors in Arbitrary Knapsack Systems, September 1982

TM-231 Kleitman, D., Leighton, F.T., Lepley, M. and Miller G.L.
An Asymptotically Optimal Layout for the Shuffle-Exchange Graph, October 1982

TM-232 Yeh, A.
PLY: A System of Plausibility Inference with a Probabilistic Basis, December 1982

TM-233 Konopelski, L.J.
Implementing Internet Remote Login on a Personal Computer, S.B. Thesis, Electrical Engineering Department, December 1982

TM-234 Rivest, R. and Sherman, A.T.
Randomized Encryption Techniques, January 1983

TM-235 Mitchell, J.C.
The Implementation of Problem for Functional and Inclusion Dependencies, February 1983

TM-236 Leighton, F.T. and Leiserson, C.E.
Wafter-Scale Integration of Systolic Arrays, February 1983

TM-237 Dolev, D., Leighton, F.T. and Trickey, H.
Planar Embedding of Planar Graphs, February 1983

TM-238 Baker, B.S., Bhatt, S.N. and Leighton, F.T.
An Approximation Algorithm for Manhattan Routing, February 1983

TM-239 Sutherland, J.B. and Sirbu, M.
Evaluation of an Office Analysis Methodology, March 1983

TM-240 Bromley, H.
A Program for Therapy of Acid-Base and Electrolyte Disorders, S.B. Thesis, Electrical Engineering Department, June 1983

TM-241 Arvind and Iannucci, R.A.
Two Fundamental Issues in Multiprocessing: The Dataflow Solution, September 1983; A134239

TM-242 Pingali, K. and Arvind.
Efficient Demand-driven Evaluation (I), September 1983; A133477

TM-243 Pingali, K. and Arvind.
Efficient Demand-driven Evaluation (II), September 1983; A133879

TM-244 Goldreich, O., Goldwasser, S. and Micali, S.
How to Construct Random Functions, November 1983

TM-245 Meyer, A.R.
Understanding Algol: The View of the Recent Convert to Denotational Semantics, October 1983

TM-246 Trakhtenbrot, B.A., Halpern, J.Y. and Meyer, A.R.
From Denotational to Operational and Axiomatic Semantics for Algol-Like Languages: An Overview, October 1983

TM-247 Leighton, T. and Lepley, M.
Probabilistic Searching in Sorted Linked Lists, November 1983

TM-248 Leighton, F.T. and Rivest, R.L.
Estimating a Probability Using Finite Memory, November 1983

TM-249 Leighton, F.T. and Rivest, R.L.
The Markov Chain Tree Theorem, December 1983

TM-250 Goldreich, O.
On Concurrent Identification Protocols, December 1983

TM-251 Dolev, D., Lynch, N.A., Pinter, S. Stark, E. and Weihl, W.
Reaching Approximate Agreement in the Presence of Faults, December 1983

TM-252 Zachos, S. and Heller, H.
On BPP, December 1983

TM-253 Chor, B., Leiserson, C., Rivest, R. and Shearer, J.
An Application of Number Theory to the Organization of Raster Graphics Memory, April 1984; A140638

TM-254 Feldmeier, D.C.
Empirical Analysis of a Token Ring Network, January 1984; A138905

TM-255 Bhatt, S. and Leiserson, C.
How to Assemble Tree Machines, March 1984; A139963

TM-256 Goldreich, O.
On the Number of Close-and Equal Pairs of Bits in a String (With Implications on the Security of RSA's L.S.B.), March 1984

TM-257 Dwork, C., Kanellakis, P.C. and Mitchell, J.C.
On the Sequential Nature of Unification, March 1984; A139939

TM-258 Halpern, J.Y., Meyer A.R. and Trakhtenbrot, B.A.
The Semantics of Local Storage, or What Makes the Free-List Free?, April 1984

TM-259 Lynch, N.A. and Fredrickson, G.N.
The Impact of Synchronous Communication on the Problem of Selecting a Leader in a Ring, April 1984

TM-260 Chor, B. and Goldreich, O.
RSA/Rabin Least Significant Bits are 1/2 + 1/poly(logN) Secure, May 1984

TM-261 Zaks, S.
Optimal Distributed Algorithms for Sorting and Ranking, May 1984

TM-262 Leighton, T. and Rosenberg, A.
Three-dimensional Circuit Layouts, June 1984; A143430

TM-263 Sirbu, M.S. and Sutherland, J.B.
Naming and Directory Issues in Message Transfer Systems, July 1984; A154727

TM-264 Sarin, S.K. and Greif, I.
Software for Interactive On-Line Conferences, July 1984; A154742

TM-265 Lundelius, J. and Lynch, N.
A New Fault-Tolerant Algorithm for Clock Synchronization, July 1984; A154771

TM-266 Chor, B. and Coan, B.A.
A Simple and Efficient Randomized Byzantine Agreement Algorithm, August 1984; A153240

TM-267 Schooler, R. and Stamos, J.W.
Proposal for a Small Scheme Implementation, October 1984; A148707

TM-268 Awerbuch, B.
Complexity of Network Synchronization, January 1985

TM-269 Fischer, M., Lynch, N.A., Burns, J. and Borodin, A.
The Colored Ticket Algorithm, August 1983; A148696

TM-270 Dwork, C., Lynch, C. and Stockmeyer, L.
Consensus in the Presence of Partial Synchrony (Preliminary Version), July 1984; A154705

TM-271 Dershowitz, N. and Zaks, S.
Patterns in Trees, January 1985

TM-272 Leighton, T.
Tight Bounds on the Complexity of Parallel Sorting, April 1985; A154726

TM-273 Berman, F., Leighton, T., Shor, P.W. and Shor, L.
Generalized Planar Matching, April 1985; A154770

TM-274 Kuipers, B.
Qualitative Simulation of Mechanisms, April 1985

TM-275 Burns, J.E. and Lynch, N.A.
The Byzantine Firing Squad Problem, April 1985; A154809

TM276] Dolev, D., Lynch., N.A., Pinter, S.S., Stark, E.W. and Weihl, W.E. Reaching Approximate Agreement in the Presence of Faults, May 1985; A156541

TM-277 Frederickson, G.N. and Lynch, N.A.
A General Lower Bound for Electing a Leader in a Ring, March 1985; A156241

TM-278 Fisch, M.J., Griffeth, N.D., Guibas, L.J. and Lynch, N.A.
Probabilistic Analysis of a Network Resource Allocation Algorithm, June 1985; A157553

TM-279 Fischer, M.J., Lynch, N.A. and Merritt, M.
Easy Impossibility Proofs for Distributed Consensus Problems, June 1985; A157402

TM-280 Kuipers, B. and Kassirer, J.P.
Qualitative Simulation in Medical Physiology: A Progress Report, June 1985

TM-281 Hailperin, M.
What Price for Eliminating Expression Side-Effects?, June 1985

TM-282 Sarin, S. and Greif, I.
Computer Based Real-Time Conferences, July 1985

TM-283 Chor, B. and Goldreich, O.
Unbiased Bits from Sources of Weak Randomness and Probabilistic Communication Complexity, September 1986

TM-284 Leiserson, C.E. and Saxe, J.B.
A Mixed-Integer Linear Programming Problems Which Is Efficiently Solvable, July 1985; A159496

TM-285 Kilian, J.J
Two Undecidability Results in Probabilistic Automata Theory, S.B. Thesis, EE & CS Department, June 1985

TM-286 Hastad, J.
Improvements of Yao's Results on Parity Circuits, September 1985

TM-287 Rivest, R.L.
Network Control by Bayesian Broadcast, July 1985

TM-288 Chung, J.C.
Dscribe: A Scribe Server, May 1985

TM-289 Toffoli, T. and Margolus, N.
The CAM-7 Multiprocessor: A Cellular Automata Machine, December 1985

TM-290 Fischer, M.J., Lynch, N.A., Burns, J.E. and Borodin, A.
Distributed FIFO Allocation of Identical Resources Using Small Shared Space, June 1985

TM-291 Goldberg, A.V.
A New Max-Flow Algorithm, November 1985

TM-292 Jain, R. and Routhier, S.
Packet Trains: Measurements and a New Model for Computer Network Traffic, November 1985

TM-293 Barrington, D.A.
Width-3 Permutation Branching Programs, December 1985

TM-294 Arvind and Culler, D.E.
Dataflow Architectures, February 1986; A166235

TM-295 Greif, I., Selinger, R. and Weihl, W.
Atomic Data Abstractions in a Distributed Collaborative Editing System (Extended Abstract), November 1985

TM-296 Margolus, N., Toffoli, T. and Vichniac, G.
Cellular Automata Supercomputers for Fluid Dynamics Modeling, December 1985

TM-297 Bentley, J.L., Leighton, F.T., Lepley, M., Stanat, D.F. and Steele, J.M. A Randomized Data Structure for Ordered Sets, May 1986

TM-298 Leighton, T. and Shor, P.
Tight Bounds for Minimax Grid Matching, with Applications to the Average Case Analysis of Algorithms, May 1986

TM-299 Gifford, D.K., Lucassen, J.M. and Berlin, S.T,
The Application of Digital Broadcast Communication to Large Scale Information Systems, April 1986

TM-300 Dwork, C. and Moses, Y.
Knowledge and Common Knowledge in a Byzantine Environment: Crash Failures, July 1986

TM-301 Elias, P.
Interval and Regency-Rank Source Coding: Two On-Line Adaptive Variable-Length Schemes, April 1986

TM-302 Leighton, T. and Leiserson, C.E.
A Survey of Algorithms for Integrating Wafer-Scale Systolic Arrays, May 1986; A171623

TM-303 Li, M., Longpre, L. and Vitanyi, P.M.B.
The Power of the Queue, April 1986

TM-304 Kranakis, E. and Vitanyi, P.M.B.
Distributed Control in Computer Networks and Cross-Sections of Colored Multidimensional Bodies, April 1986; A172224

TM-305 Sacks, E.
Representing Change, May 1986

TM-306 Vitanyi, P.M.B.
Nonsequential Computation and Laws of Nature, May 1986; A171505

TM-307 Greenberg, R.I. and Leiserson, C.E.
Randomized Routing on Fat-Trees, April 1986

TM-308 Meyer, A.R.
Floyd-Hoare Logic Defines Semantics, May 1986

TM-309 Leiserson, C.E. and Saxe, J.B.
Retiming Synchronous Circuitry, May 1986; A172144

TM-310 Szolovits, P. Kassirer, J.P., Long, W.J., Moskowitz, A.J., Pauker, S.G., Patil, R.S. and Wellman, M.P.
An Artificial Intelligence Approach to Clinical Decision Making, September 1986

TM-311 Rivest, R.L.
Game Tree Searching By Min/Max Approximation, September 1986

TM-312 Sacks, E.
Hierarchical Inequality Reasoning, February 1987

TM-313 Theory of Computation Research Group
Theory of Computation Research Group Summary, 1985-86, August 1986

TM-314 Vitanyi, P.M.B. and Awerbuch, B.
Atomic Shared Register Access By Asynchronous Hardware (Detailed Abstract), October 1986; A178301

TM-315 Goldreich, O.
Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme, September 1986

TM-316 Greif, I. and Sarin, S.
Data Sharing in Group Work, October 1986

TM-317 Bennett, C.H., Toffoli, T. and Wolfram, S.
Cellular Automata '86 Conference, December 1986

TM-318 Leiserson, C.E. and Maggs, B.M.
Communication-Efficient Parallel Graph Algorithms, December 1986

TM-319 Leong, T-Y.
Murmur Clinic: An Auscultation Expert System, January 1987

TM-320 Goldberg, A.V. and Plotkin, S.A.
Efficient Parallel Algorithms for $(\Delta + 1)$-Coloring and Maximal Independent Set Problems, February 1987

TM-321 Cormen, T.H. and Leiserson, C.E.
A Hyperconcentrator Switch for Routing Bit-Serial Messages, February 1987; A17840

TM-322 Cormen, T.H.
Efficient Multichip Partial Concentrator Switches, February 1987; A178334

TM-323 Leiserson, C.E. and Phillips, C.A.
A Space-Efficient Algorithm for Finding the Connected Components of Rectangles in the Plane, February 1987

TM-324 Fekete, A. Lynch, N., Merritt, M. and Weihl, W.
Nested Transactions and Read/Write Locking, April 1987; A191981

TM-325 Awerbuch, B., Goldreich, O. and Vainish, R.
On the Message Complexity of Broadcast: A Basic Lower Bound (Extended Abstract), May 1987

TM-326 Awerbuch, B.
Controlling Worst-Case Performance of a Communication Protocol and Dynamic Resource Management, May 1987

TM-327 Awerbuch, B.
Adapting Communication Protocols to Dynamic Input and Network Topology: Research Summary, May 1987

TM-328 Awerbuch, B. and Plotkin, S.A.
Approximating the Size of a Dynamically Growing Asynchronous Distributed Network, April 1987

TM-329 Herlihy, M., Lynch, N., Merritt, M. and Weihl, W.
On the Correctness of Orphan Elimination Algorithms; May 1987; A182175

TM-330 Arvind and Iannucci, R.A.
Two Fundamental Issues in Multiprocessing (replaces TM-241), October 1987

TM-331 Russ, T.A.
Temporal Control Structure Reference Manual, June 1987

TM-332 Wellman, M.P.
Formulation of Tradeoffs in Planning Under Uncertainty, June 1987

TM-333 Goldberg, A.V. and Tarjan, R.E.
Finding Minimum-Cost Circulations by Successive Approximation, July 1987

TM-334 Goldberg, A.V. and Tarjan, R.E.
Finding Minimum-Cost Circulations by Canceling Negative Cycles, July 1987

TM-335 Bruce, K.B. and Riecke, J.G.
The Semantics of Miranda's Algebraic Types, August 1987

TM-336 Beame, P.
Lower Bounds for Recognizing Small Cliques on CROW PRAMS, August 1987

TM-337 Theory of Computation Research Group
Theory of Computation Group Research Summary June 1986-87, August 1987

TM-338 Fujita, T.
Multithreaded Processor Architecture for Parallel Symbolic Computation, September 1987

TM-339 Quinlan, J.R. and Rivest, R.L.
Inferring Decision Trees Using the Minimum Description Length Principle, September 1987

TM-340 Fekete, A., Lynch, N., Merritt, M. and Weihl, W.
Nested Transactions, Conflict-Based Locking and Dynamic Atomicity, September 1987

TM-341 Fekete, A., Lynch, N. and Shrira, L.
A Modular Proof of Correctness for a Network Synchronizer, September 1987

TM-342 Fekete, A.
Approximate Agreement, September 1987

TM-343 Leiserson, C.E. and Saxe, J.B.
A Mixed-Integer Linear Programming Problem, October 1987

TM-344 Meyer, A.R. and Sieber, K.
Towards Fully Abstract Semantics for Local Variables: Preliminary Report, November 1987

TM-345 Bloom, B., Istrail, S. and Meyer, A.R.
Bisimulation Can't Be Traced: Preliminary Report, November 1987

TM-346 Awerbuch, B.
Space Efficient Dynamic Protocols: Part I, December 1987

TM-347 Awerbuch, B.
Space Efficient Dynamic Protocols: Part II, December 1987

TM-348 Haimowitz, I.J.
Using NIKL in Large Medical Knowledge Base, December 1987

TM-349 Lynch, N.A. and Stark, E.W.
A proof of the Kahn principle for Input/Output Automata, January 1988

TM-350 Awerbuch, B.
Linear Time Algorithm for Minimum Network Partition, March 1988

TM-351 Lynch, N.
I/O Automata: A Model for Discrete Event Systems, March 1988

TM-352 Feldmeier, D.C.
Estimated Performance of a Gateway Routing Table Cache, March 1988

TM-353 Meyer, A.R.
Semantical Paradigms: Notes for an Invited Lecture, July 1988

TM-354 Awerbuch, B., Bar-Noy, A., Linial, N., and Peleg, D.
Improved Routing Strategies with Succinct Tables, April 1988

TM-355 Lynch, N., Mansour, Y. and Fekete, A.
The Data Link Layer: Two Impossibility Results, May 1988

TM-356 Goldberg, A.V., Plotkin, S.A. and Shannon, G.E.
Parallel Symmetry-Breaking in Sparse Graphs, May 1988

TM-357 Goldberg, A.V., Plotkin, S.A. and Vaidya, P.
Sublinear-Time Parallel Algorithms for Matching and Related Problems, July 1988

TM-358 Goldberg, A.V., Plotkin, S.A. and Tardos, E.
Combinatorial Algorithms for the Generalized Circulation Problem, May 1988

TM-359 Welch, J.L.
Simulating Synchronous Processors, June 1988

TM-360 Coan, B.A. and Welch, J.L.
Transaction Commit in a Realistic Timing Model, June 1988

TM-361 Welch, J.L., Lamport, L. and Lynch, N.
A Lattice-Structured Proof Technique Applied to a Minimum Spanning Tree Algorithm, June 1988

TM-362 Lynch, N., Merritt, M., Weihl, W.E. and Fekete, A.
A Theory of Atomic Transaction, June 1988

TM-363 Fogg, D.C.
Assisting Design Given Multiple Performance Criteria, August 1988

TM-364 Schaffer, R. and Bloom, B. (editor)
On the Correctness of Atomic Multi-Writer Registers, June 1988

TM-365 Awerbuch, B., Goldreich, O., Peleg, D. and Vainish, R.
A Tradeoff Between Information and Communication in Broadcast Protocols, July 1988

TM-366 Goldman, S.A.
A Space Efficient Greedy Triangulation Algorithm, July 1988

TM-367 Weihl, W.E.
Commutativity-Based Concurrency Control for Abstract Data Types, August 1988

TM-368 Herlihy, M.P. and Weihl, W.E.
Hybrid Concurrency Control For Abstract Data Types, August 1988

TM-369 Awerbuch, B., Bar-Noy, A., Linial, N. and Peleg, D.
Memory-Balanced Routing Strategies, August 1988

TM-370 Fekete, A.,Lynch, N., Merritt, M. and Weihl, B.
Commutativity-Based Locking for Nesting Transactions, August 1988

TM-371 Gifford, D.
Natural Random Numbers, September 1988

TM-372 Leiserson, C.E. and Saxe, J.B.
Retiming Synchronous Circuitry, October 1988

TM-373 Lynch, N.A. and Tuttle, M.R.
An Introduction to Input/Output Automata, November 1988

TM-374 Grigni, M. and Peleg, D.
Tight Bounds on Minimum Broadcast Network, October 1988

TM-375 Awerbuch, B.
On the Effects of Feedback in Dynamic Network Protocols, December 1988

TM-376 Awerbuch, B., Bar-Noy, A., Linial, N. and Peleg, D.
Improved Routing Strategies with Succinct Tables, December 1988

TM-377 TOC
Theory of Computation Group Research Summary June 1987-June 1988, January 1989

TM-378 Jouvelot, P. and Gifford, D.K.
Reasoning about Continuations with Control Effects, January 1989

TM-379 Kilian, J., Kipnis, S. and Leiserson, C.E.
The Organization of Permutation Architecture with Bussed Interconnections, January 1989

TM-380 O'Toole, J.W. and Gifford, D.K.
Type Reconstruction with First Class Polymorphic Values, May 1989

TM-381 Elias, P.
Error-Correcting Code for List Decoding, February 1989

TM-382 Weihl, W.
The Impact of Recovery on Concurrency Control, February 1989

TM-383 Wang, P.
A Special Case of Second-Order Strictness Analysis, February 1989

TM-384 Lamport, L. and Lynch, N.A.
Chapter on Distributed Computing, February 1989

TM-385 Sloan, R.
All Zero-Knowledge Proofs are Proofs of Language Membership, February 1989

TM-386 Jouvelot, P. and Gifford, D.K.
Communication Effects for Message-Based Concurrency, February 1989

TM-387 Doyle, J. and Patil, R.S.
Language Restrictions, Taxonomic Classification, and the Utility of Representation Services, May 1989

TM-388 Kipnis, S.
Three Methods for Range Queries in Computational Geometry, March 1989

TM-389 Afek, Y., Awerbuch, B. and Moriel, H.
A Complexity Preserving Reset Procedure, May 1989

TM-391 Awerbuch, B., Mansour, Y. and Shavit, N.
Polynomial End-to-End Communication, May 1989, A213 776

TM-392 Attiya, H., Dolev, D. and Shavit, N.
Bounded Polynomial Randomized Consensus, June 1989, A213 808

TM-393 Dolev, D. and Shavit, N.
Bounded Concurrent Time-Stamp Systems are Constructible, June 1989, A213 053

TM-394 Lynch, N.
A Hundred Impossibility Proofs for Distributed Computing, August 1989

TM-395 Owicki, S. and Anant, A.
Evaluating the Performance of Software Cache Coherence, June 1989

TM-396 Agarwal, A. and Cherian, M.
Adaptive Backoff Synchronization Techniques, June 1989, A213 966

TM-397 Agarwal, A. and Gupta, A.
Temporal, Processor, and Spatial Locality in Multiprocessor Memory References, June 1989, A213 790

TM-398 Goldman, K.J.
Paralation Views: Abstractions for Efficient Scientific Computing on the Connection Machine, June 1989

TM-399 Colby, C.P.
Correctness Proofs of the Peterson-Fischer Mutual Exclusion Algorithm, S.B. Thesis, June 1989

TM-400 Nour, M.F.
An Automata-Theoretic Model for Unity, S.B. Thesis, June 1989

Technical Reports[2]

TR-1 Bobrow, D.G.
Natural Language Input for a Computer Problem Solving System, Ph.D. Dissertation, Math. Department, September 1964; AD 604730

TR-2 Raphael, B.
SIR: A Computer Program for Semantic Information Retrieval, Ph.D. Dissertation, Math. Department, June 1964; AD 608499

TR-3 Corbato, F.J.
System Requirements for Multiple-Access, Time-Shared Computers, May 1964; AD 608501

TR-4 Ross, D.T. and Feldman, C.G.
Verbal and Graphical Language for the AED System: A Progress Report, May 1964; AD 604678

TR-6 Biggs, J.M. and Logcher, R.D.
STRESS: A Problem-Oriented Language for Structural Engineering, May 1964; AD 604679

TR-7 Weizenbaum, J.
OPL-1: An Open Ended Programming System within CTSS, April 1964; AD 604680

TR-8 Greenberger, M.
The OPS-1 Manual, May 1964; AD 604681

TR-11 Dennis, J.B.
Program Structure in a Multi-Access Computer, May 1964; AD 604500

TR-12 Fano, R.M.
The MAC System: A Progress Report, October 1964; AD 609296

TR-13 Greenberger, M.
A New Methodology for Computer Simulation, October 1964; AD 609288

TR-14 Roos, D.
Use of CTSS in a Teaching Environment, November 1964; AD 661807

TR-16 Saltzer, J.H.
CTSS Technical Notes, March 1965; AD 612702

TR-17 Samuel, A.L.
Time-Sharing on a Multiconsole Computer, March 1965; AD 462158

TR-18 Scherr, A.L.
An Analysis of Time-Shared Computer Systems, Ph.D. Dissertation, EE Department, June 1965; AD 470715

[2]TRs 5, 9, 10, 15 were never issued

TR-19 Russo, F.J.
A Heuristic Approach to Alternate Routing in a Job Shop, S.B. & S.M. Thesis, Sloan School, June 1965; AD 474018

TR-20 Wantman, M.E.
CALCULAID: An On-Line System for Algebraic Computation and Analysis, S.M. Thesis, Sloan School, September 1965; AD 474019

TR-21 Denning, P.J.
Queueing Models for File Memory Operation, S.M. Thesis, EE Department, October 1965; AD 624943

TR-22 Greenberger, M.
The Priority Problem, November 1965; AD 625728

TR-23 Dennis, J.B. and Van Horn, E.C.
Programming Semantics for Multi-programmed Computations, December 1965; AD 627537

TR-24 Kaplow, R., Strong, S. and Brackett, J.
MAP: A System for On-Line Mathematical Analysis, January 1966; AD 476443

TR-25 Stratton, W.D.
Investigation of an Analog Technique to Decrease Pen-Tracking Time in Computer Display, S.M. Thesis, EE Department, March 1966; AD 631396

TR-26 Cheek, T.B.
Design of a Low-Cost Character Generator for Remote Computer Displays, S.M. Thesis, EE Department, March 1966; AD 631269

TR-27 Edwards, D.J.
OCAS - On-Line Cryptanalytic Aid System, S.M. Thesis, EE Department, May 1966; AD 633678

TR-28 Smith, A.A.
Input/Output in Time-Shared, Segmented, Multiprocessor Systems, S.M. Thesis, EE Department, June 1966; AD 637215

TR-29 Ivie, E.L.
Search Procedures Based on Measures of Relatedness between Documents, Ph.D. Dissertation, EE Department, June 1966; AD 636275

TR-30 Saltzer, J.H.
Traffic Control in a Multiplexed Computer System, Sc.D. Thesis, EE Department, July 1966; AD 635966

TR-31 Smith, D.L.
Models and Data Structures for Digital Logic Simulation, S.M. Thesis, EE Department, August 1966; AD 637192

TR-32 Teitelman, W.
PILOT: A Step Toward Man-Computer Symbiosis, Ph.D. Dissertation, Math. Department, September 1966; AD 638446

TR-33 Norton, L.M.
ADEPT - A Heuristic Program for Proving Theorems of Group Theory, Ph.D. Dissertation, Math. Department, October 1966; AD 645660

TR-34 Van Horn, E.C.
Computer Design for Asynchronously Reproducible Multiprocessing, Ph.D. Dissertation, EE Department, November 1966; AD 650407

TR-35 Fenichel, R.R.
An On-Line System for Algebraic Manipulation, Ph.D. Dissertation, Applied Mathematics (Harvard), December 1966; AD 657282

TR-36 Martin, W.A.
Symbolic Mathematical Laboratory, Ph.D. Dissertation, EE Department, January 1967; AD 657283

TR-37 Guzman-Arenas, A.
Some Aspects of Pattern Recognition by Computer, S.M. Thesis, EE Department, February 1967; AD 656041

TR-38 Rosenberg, R.C., Kennedy, D.W. and Humphrey, R.A.
A Low-Cost Output Terminal For Time-Shared Computers, March 1967; AD 662027

TR-39 Forte, A.
Syntax-Based Analytic Reading of Musical Scores, April 1967; AD 661806

TR-40 Miller, J.R.
On-Line Analysis for Social Scientists, May 1967; AD 668009

TR-41 Coons, S.A.
Surfaces for Computer-Aided Design of Space Forms, June 1967; AD 663504

TR-42 Liu, C.L., Chang, G.D. and Marks, R.E.
Design and Implementation of a Table-Driven Compiler System, July 1967; AD 668960

TR-43 Wilde, D.U.
Program Analysis by Dig[illegible] Computer, Ph.D. Dissertation, EE Department, August 1967; AD 662224

TR-44 Gorry, G.A.
A System for Computer-Aided Diagnosis, Ph.D. Dissertation, Sloan School, September 1967; AD 662665

TR-45 Leal-Cantu, N.
On the Simulation of Dynamic Systems with Lumped Parameters and Time Delays, S.M. Thesis, ME Department, October 1967; AD 663502

TR-46 Alsop, J.W.
A Canonic Translator, S.B. Thesis, EE Department, November 1967; AD 663503

TR-47 Moses, J.
Symbolic Integration, Ph.D. Dissertation, Math. Department, December 1967; AD 662666

TR-48 Jones, M.M.
Incremental Simulation on a Time-Shared Computer, Ph.D. Dissertation, Sloan School, January 1968; AD 662225

TR-49 Luconi, F.L.
Asynchronous Computational Structures, Ph.D. Dissertation, EE Department, February 1968; AD 667602

TR-50 Denning, P.J.
Resource Allocation in Multiprocess Computer Systems, Ph.D. Dissertation, EE Department, May 1968; AD 675554

TR-51 Charniak, E.
CARPS, A Program which Solves Calculus Word Problems, S.M. Thesis, EE Department, July 1968; AD 673670

TR-52 Deitel, H.M.
Absentee Computations in a Multiple-Access Computer System, S.M. Thesis, EE Department, August 1968; AD 684738

TR-53 Slutz, D.R.
The Flow Graph Schemata Model of Parallel Computation, Ph.D. Dissertation, EE Department, September 1968; AD 683393

TR-54 Grochow, J.M.
The Graphic Display as an Aid in the Monitoring of a Time-Shared Computer System, S.M. Thesis, EE Department, October 1968; AD 689468

TR-55 Rappaport, R.L.
Implementing Multi-Process Primitives in a Multiplexed Computer System, S.M. Thesis, EE Department, November 1968; AD 689469

TR-56 Thornhill, D., Stotz, R.H., Ross, D.T. and Ward, J.E.
An Integrated Hardware-Software System for Computer Graphics in Time-Sharing, December 1968; AD 685202

TR-57 Morris, J.H.
Lambda Calculus Models of Programming Languages, Ph.D. Dissertation, Sloan School, December 1968; AD 683394

TR-58 Greenbaum, H.J.
A Simulator of Multiple Interactive Users to Drive a Time-Shared Computer System, S.M. Thesis, EE Department, January 1969; AD 686988

TR-59 Guzman-Arenas, A.
Computer Recognition of Three-Dimensional Objects in a Visual Scene, Ph.D. Dissertation, EE Department, December 1968; AD 692200

TR-60 Ledgard, H.F.
A Formal System for Defining the Syntax and Semantics of Computer Languages, Ph.D. Dissertation, EE Department, April 1969; AD 689305

TR-61 Baecker, R.M.
Interactive Computer-Mediated Animation, Ph.D. Dissertation, EE Department, June 1969; AD 690887

TR-62 Tillman, C.C.
EPS: An Interactive System for Solving Elliptic Boundary-Value Problems with Facilities for Data Manipulation and General-Purpose Computation, June 1969; AD 692462

TR-63 Brackett, J., Hammer, M.M. and Thornhill, D.
Case Study in Interactive Graphics Programming: A Circuit Drawing and Editing Program for Use with a Storage-Tube Display Terminal, October 1969; AD 699930

TR-64 Rodrigues, J.E.
A Graph Model for Parallel Computations, Sc.D. Thesis, EE Department, September 1969; AD 697759

TR-65 Deremer, F.L.
Practical Translators for LR(k) Languages, Ph.D. Dissertation, EE Department, October 1969; AD 699501

TR-66 Beyer, W.T.
Recognition of Topological Invariants by Iterative Arrays, Ph.D. Dissertation, Math Department, October 1969; AD 699502

TR-67 Vanderbilt, D.H.
Controlled Information Sharing in a Computer Utility, Ph.D. Dissertation, EE Department, October 1969; AD 699503

TR-68 Selwyn, L.
Economies of Scale in Computer Use: Initial Tests and Implications for The Computer Utility, Ph.D. Dissertation, Sloan School, June 1970; AD 710001

TR-69 Gertz, J.L.
Hierarchical Associative Memories for Parallel Computation, Ph.D. Dissertation, EE Department, June 1970; AD 711091

TR-70 Fillat, A.I. and Kraning, L.A.
Generalized Organization of Large Data Bases: A Set-Theoretic Approach to Relations, S.B. & S.M. Thesis, EE Department, June 1970; AD 711060

TR-71 Fiasconaro, J.G.
A Computer-Controlled Graphical Display Processor, S.M. Thesis, EE Department, June 1970; AD 710479

TR-72 Patil, S.S.
Coordination of Asynchronous Events, Sc.D. Thesis, EE Department, June 1970; AD 711763

TR-73 Griffith, A.K.
Computer Recognition of Prismatic Solids, Ph.D. Dissertation, Math. Department, August 1970; AD 712069

TR-74 Edelberg, M.
Integral Convex Polyhedra and an Approach to Integralization, Ph.D. Dissertation, EE Department, August 1970; AD 712070

TR-75 Hebalkar, P.G.
Deadlock-Free Sharing of Resources in Asynchronous Systems, Sc.D. Thesis, EE Department, September 1970; AD 713-139

TR-76 Winston, P.H
Learning Structural Description ¿from Examples, Ph.D. Dissertation, EE Department, September 1970; AD 713988

TR-77 Haggerty, J.P.
Complexity Measures for Language Recogniti n by Canonic Systems, S.M. Thesis, EE Department, October 1970; AD 715134

TR-78 Madnick, S.E.
Design Strategies for File Systems, S.M. Thesis, EE Department & Sloan School, October 1970; AD 714269

TR-79 Horn, B.K.P.
Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View, Ph.D. Dissertation, EE Department, November 1970; AD 717336

TR-80 Clark, D.D., Graham, R.M., Saltzer, J.H. and Schroeder, M.D.
The Classroom Information and Computing Service, January 1971; AD 717857

TR-81 Banks, E.R.
Information Processing and Transmission in Cellular Automata, Ph.D. Dissertation, ME Department, January 1971; AD 717951 w

TR-82 Krakauer, L.J.
Computer Analysis of Visual Properties of Curved Objects, Ph.D. Dissertation, EE Department, May 1971; AD 723647

TR-83 Lewin, D.
In-Process Manufacturing Quality Control, Ph.D. Dissertation, Sloan School, January 1971; AD 720098

TR-84 Winograd, T.
Procedures as a Representation for Data in a Computer Program for Understanding Natural Language, Ph.D. Dissertation, Math. Department, February 1971; AD 721399

TR-85 Miller, P.L.
Automatic Creation of a Code Generator ¿from a Machine Description, E.E. Thesis, EE Department, May 1971; AD 724730

TR-86 Schell, R.R.
Dynamic Reconfiguration in a Modular Computer System, Ph.D. Dissertation, EE Department, June 1971; AD 725859

TR-87 Thomas, R.H.
A Model for Process Representation and Synthesis, Ph.D. Dissertation, EE Department, June 1971; AD 726049

TR-88 Welch, T.A.
Bounds on Information Retrieval Efficiency in Static File Structures, Ph.D. Dissertation, EE Department, June 1971; AD 725429

TR-89 Owens, R.C.
Primary Access Control in Large-Scale Time-Shared Decision Systems, S.M. Thesis, Sloan School, July 1971; AD 728036

TR-90 Lester, B.P.
Cost Analysis of Debugging Systems, S.B. & S.M. Thesis, EE Department, September 1971; AD 730521

TR-91 Smoliar, S.W.
A Parallel Processing Model of Musical Structures, Ph.D. Dissertation, Math. Department, September 1971; AD 731690

TR-92 Wang, P.S.
Evaluation of Definite Integrals by Symbolic Manipulation, Ph.D. Dissertation, Math. Department, October 1971; AD 732005

TR-93 Greif, I.
Induction in Proofs about Programs, S.M. Thesis, EE Department, February 1972; AD 737701

TR-94 Hack, M.
Analysis of Production Schemata by Petri Nets, S.M. Thesis, EE Department, February 1972; AD 740320

TR-95 Fateman, R.J.
Essays in Algebraic Simplification (A revision of a Harvard Ph.D. Dissertation), April 1972; AD 740132

TR-96 Manning, F.
Autonomous, Synchronous Counters Constructed Only of J-K Flip-Flops, S.M. Thesis, EE Department, May 1972; AD 744030

TR-97 Vilfan, B.
The Complexity of Finite Functions, Ph.D. Dissertation, EE Department, March 1972; AD 739678

TR-98 Stockmeyer, L.J.
Bounds on Polynomial Evaluation Algorithms, S.M. Thesis, EE Department, April 1972; AD 740328

TR-99 Lynch, N.A.
Relativization of the Theory of Computational Complexity, Ph.D. Dissertation, Math. Department, June 1972; AD 744032

TR-100 Mandl, R.
Further Results on Hierarchies of Canonic Systems, S.M. Thesis, EE Department, June 1972; AD 744206

TR-101 Dennis, J.B.
On the Design and Specification of a Common Base Language, June 1972; AD 744207

TR-102 Hossley, R.F.
Finite Tree Automata and Ω-Automata, S.M. Thesis, EE Department, September 1972; AD 749367

TR-103 Sekino, A.
Performance Evaluation of Multiprogrammed Time-Shared Computer Systems, Ph.D. Dissertation, EE Department, September 1972; AD 749949

TR-104 Schroeder, M.D.
Cooperation of Mutually Suspicious Subsystems in a Computer Utility, Ph.D. Dissertation, EE Department, September 1972; AD 750173

TR-105 Smith, B.J.
An Analysis of Sorting Networks, Sc.D. Thesis, EE Department, October 1972; AD 751614

TR-106 Rackoff, C.
The Emptiness and Complementation Problems for Automata on Infinite Trees, S.M. Thesis, EE Department, January 1973; AD 756248

TR-107 Madnick, S.E.
Storage Hierarchy Systems, Ph.D. Dissertation, EE Department, April 1973; AD 760001

TR-108 Wand, M.
Mathematical Foundations of Formal Language Theory, Ph.D. Dissertation, Math. Department, December 1973.

TR-109 Johnson, D.S.
Near-Optimal Bin Packing Algorithms, Ph.D. Dissertation, Math. Department, June 1973, PB 222-090

TR-110 Moll, R.
Complexity Classes of Recursive Functions, Ph.D. Dissertation, Math. Department, June 1973; AD 767730

TR-111 Linderman, J.P.
Productivity in Parallel Computation Schemata, Ph.D. Dissertation, EE Department, December 1973, PB 226-159/AS

TR-112 Hawryszkiewycz, I.T.
Semantics of Data Base Systems, Ph.D. Dissertation, EE Department, December 1973, PB 226-061/AS

TR-113 Herrmann, P.P.
On Reducibility Among Combinatorial Problems, S.M. Thesis, Math. Department, December 1973, PB 226-157/AS

TR-114 Metcalfe, R.M.
Packet Communication, Ph.D. Dissertation, Applied Math., Harvard University, December 1973; AD 771430

TR-115 Rotenberg, L.
Making Computers Keep Secrets, Ph.D. Dissertation, EE Department, February 1974, PB 229-352/AS

TR-116 Stern, J.A.
Backup and Recovery of On-Line Information in a Computer Utility, S.M. & E.E. Thesis, EE Department, January 1974; AD 774141

TR-117 Clark, D.D.
An Input/Output Architecture for Virtual Memory Computer Systems, Ph.D. Dissertation, EE Department, January 1974; AD 774738

TR-118 Briabrin, V.
An Abstract Model of a Research Institute: Simple Automatic Programming Approach, March 1974, PB 231-505/AS

TR-119 Hammer, M.M.
A New Grammatical Transformation into Deterministic Top-Down Form, Ph.D. Dissertation, EE Department, February 1974; AD 775545

TR-120 Ramchandani, C.
Analysis of Asynchronous Concurrent Systems by Timed Petri Nets, Ph.D. Dissertation, EE Department, F.bruary 1974; AD 775618

TR-121 Yao, F.F.
On Lower Bounds for Selection Problems, Ph.D. Dissertation, Math. Department, March 1974, PB 230-950/AS

TR-122 Scherf, J.A.
Computer and Data Security: A Comprehensive Annotated Bibliography, S.M. Thesis, Sloan School, January 1974; AD 775546

TR-123 Saltzer, et al.
Introduction to Multics, February 1974; AD 918562

TR-124 Laventhal, M.S.
Verification of Programs Operating on Structured Data, S.B. & S.M. Thesis, EE Department, March 1974, PB 231-365/AS

TR-125 Mark, W.S.
A Model-Debugging System, S.B. & S.M. Thesis, EE Department, April 1974; AD 778688

TR-126 Altman, V.E.
A Language Implementation System, S.B. & S.M. Thesis, Sloan School, May 1974; AD 780672

TR-127 Greenberg, B.
An Experimental Analysis of Program Reference Patterns in the Multics Virtual Memory, S.M. Thesis, EE Department, May 1974; AD 780407

TR-128 Frankston, R.M.
The Computer Utility as a Marketplace for Computer Services, S.M. & E.E. Thesis, EE Department, May 1974; AD 780436

TR-129 Weissberg, R.
Using Interactive Graphics in Simulating the Hospital Emergency Room, S.M. Thesis, EE Department, May 1974; AD 780437

TR-130 Ruth, G.R.
Analysis of Algorithm Implementations, Ph.D. Dissertation, EE Department, May 1974; AD 780408

TR-131 Levin, M.
Mathematical Logic for Computer Scientists, June 1974.

TR-132 Janson, P.A.
Removing the Dynamic Linker from the Security Kernel of a Computing Utility, S.M. Thesis, EE Department, June 1974; AD 781305

TR-133 Stockmeyer, L.J.
The Complexity of Decision Problems in Automata Theory and Logic, Ph.D. Dissertation, EE Department, July 1974, PB 235-283/AS

TR-134 Ellis, D.J.
Semantics of Data Structures and References, S.M. & E.E. Thesis, EE Department, August 1974, PB 236-594/AS

TR-135 Pfister, G.F.
The Computer Control of Changing Pictures, Ph.D. Dissertation, EE Department, September 1974; AD 787795

TR-136 Ward, S.
Functional Domains of Applicative Languages, Ph.D. Dissertation, EE Department, September 1974; AD 787796

TR-137 Seiferas, J.
Nondeterministic Time and Space Complexity Classes, Ph.D. Dissertation, Math. Department, September 1974; PB 236777/AS

TR-138 Yun, David Y.Y.
The Hensel Lemma in Algebraic Manipulation, Ph.D. Dissertation, Math. Department, November 1974; A 002737

TR-139 Ferrante, J.
Some Upper and Lower Bounds on Decision Procedures in Logic, Ph.D. Dissertation, Math. Department, November 1974. PB 238121/AS

TR-140 Redell, D.D.
Naming and Protection in Extendable Operating Systems, Ph.D. Dissertation, EE Department, November 1974; A 001721

TR-141 Richards, M., Evans, A. Jr. and Mabee, R.F.
The BCPL Reference Manual, December 1974; A 003599

TR-142 Brown, G.P.
Some Problems in German to English Machine Translation, S.M. & E.E. Thesis, EE Department, December 1974; A 003002

TR-143 Silverman, H.
A Digitalis Therapy Advisor, S.M. Thesis, EE Department, January 1975.

TR-144 Rackoff, C.
The Computational Complexity of Some Logical Theories, Ph.D. Dissertation, EE&CS Department, February 1975.

TR-145 Henderson, D.A.
The Binding Model: A Semantic Base for Modular Programming Systems, Ph.D. Dissertation, EE&CS Department, February 1975; AD A006961

TR-146 Malhotra, A.
Design Criteria for a Knowledge-Based English Language System for Management: An Experimental Analysis, Ph.D. Dissertation, EE&CS Department, February 1975.

TR-147 Van De Vanter, M.L.
A Formalization and Correctness Proof of the CGOL Language System, S.M. Thesis, EE&CS Department, March 1975.

TR-148 Johnson, J.
Program Restructuring for Virtual Memory Systems, Ph.D. Dissertation, EE&CS Department, March 1975; A009218

TR-149 Snyder, A.
A Portable Compiler for the Language C, S.B. & S.M. Thesis, EE&CS Department, May 1975; A010218

TR-150 Rumbaugh, J.E.
A Parallel Asynchronous Computer Architecture for Data Flow Programs, Ph.D. Dissertation, EE&CS Department, May 1975; A010918

TR-151 Manning, F.
Automatic Test, Configuration and Repair of Cellular Arrays, Ph.D. Dissertation, EE&CS Department, June 1975; A012822

TR-152 Qualitz, J.E.
Equivalence Problems for Monadic Schemas, Ph.D. Dissertation, EE&CS Department, June 1975; A012823

TR-153 Miller, P.B.
Strategy Selection in Medical Diagnosis, S.M. Thesis, EE & CS Department, September 1975.

TR-154 Greif, I.
Semantics of Communicating Parallel Processes, Ph.D. Dissertation, EE & CS Department, September 1975; A016302

TR-155 Kahn, K.M.
Mechanization of Temporal Knowledge, S.M. Thesis, EE & CS Department, September 1975.

TR-156 Bratt, R.G.
Minimizing the Naming Facilities Requiring Protection in a Computing Utility, S.M. Thesis, EE & CS Department, September 1975.

TR-157 Meldman, J.A.
A Preliminary Study in Computer-Aided Legal Analysis, Ph.D. Dissertation, EE & CS Department. November 1975; A018997

TR-158 Grossman, R.W.
Some Data Base Applications of Constraint Expressions, S.M. Thesis, EE & CS Department, February 1976; A024-149

TR-159 Hack, M.
Petri Net Languages, March 1976.

TR-160 Bosyj, M.
A Program for the Design of Procurement Systems, S.M. Thesis, EE & CS Department, May 1976; A026-688

TR-161 Hack, M.
Decidability Questions for Petri Nets, Ph.D. Dissertation, EE & CS Department, June 1976.

TR-162 Kent, S.
Encryption-Based Protection Protocols for Interactive User-Computer Communication, S.M. Thesis, EE & CS Department, June 1976; A026-911

TR-163 Montgomery, W.A.
A Secure and Flexible Model of Process Initiation for a Computer Utility, S.M. & E.E. Thesis, EE & CS Department, June 1976.

TR-164 Reed, D.P.
Processor Multiplexing in a Layered Operating System, S.M. Thesis, EE & CS Department, July 1976.

TR-165 McLeod, D.
High Level Expression of Semantic Integrity Specifications in a Relational Data Base System, S.M. Thesis, EE & CS Department, September 1976; A034-184

TR-166 Chan, A.Y.
Index Selection in a Self-Adaptive Relational Data Base Management System, S.M. Thesis, EE & CS Department, September 1976; A034-185

TR-167 Janson, P.A.
Using Type Extension to Organize Virtual Memory Mechanisms, Ph.D. Dissertation, EE & CS Department, September 1976.

TR-168 Pratt, V.R.
Semantical Considerations on Floyd-Hoare Logic, September 1976.

TR-169 Safran, C., Desforges, J.F. and Tsichlis, P.N.
Diagnostic Planning and Cancer Management, September 1976.

TR-170 Furtek, F.C.
The Logic of Systems, Ph.D. Dissertation, EE & CS Department, December 1976.

TR-171 Huber, A.H.
A Multi-Process Design of a Paging System, S.M. & E.E. Thesis, EE & CS Department, December 1976.

TR-172 Mark, W.S.
The Reformulation Model of Expertise, Ph.D. Dissertation, EE & CS Department, December 1976; A035-397

TR-173 Goodman, N.
Coordination of Parallel Processes in the Actor Model of Computation, S.M. Thesis, EE & CS Department, December 1976.

TR-174 Hunt, D.H.
A Case Study of Intermodule Dependencies in a Virtual Memory Subsystem, S.M. & E.E. Thesis, EE & CS Department, December 1976.

TR-175 Goldberg, H.J.
A Robust Environment for Program Development, S.M. Thesis, EE & CS Department, February 1977.

TR-176 Swartout, W.R.
A Digitalis Therapy Advisor with Explanations, S.M. Thesis, EE & CS Department, February 1977.

TR-177 Mason, A.H.
A Layered Virtual Memory Manager, S.M. & E.E. Thesis, EE & CS Department, May 1977.

TR-178 Bishop, P.B.
Computer Systems with a Very Large Address Space and Garbage Collection, Ph.D. Dissertation, EE & CS Department, May 1977; A040-601

TR-179 Karger, P.A.
Non-Discretionary Access Control for Decentralized Computing Systems, S.M. Thesis, EE & CS Department, May 1977; A040-804

TR-180 Luniewski, A.
A Simple and Flexible System Initialization Mechanism, S.M. & E.E. Thesis, EE & CS Department, May 1977.

TR-181 Mayr, E.W.
The Complexity of the Finite Containment Problem for Petri Nets, S.M. Thesis, EE & CS Department, June 1977 .

TR-182 Brown, G.P.
A Framework for Processing Dialogue, June 1977; A042-370

TR-183 Jaffe, J.M.
Semilinear Sets and Applications, S.M. Thesis, EE & CS Department, July 1977.

TR-184 Levine, P.H.
Facilitating Interprocess Communication in a Heterogeneous Network Environment, S.B. & S.M. Thesis, EE & CS Department, July 1977; A043-901

TR-185 Goldman, B.
Deadlock Detection in Computer Networks, S.B. & S.M. Thesis, EE & CS Department, September 1977; A047-025

TR-186 Ackerman, W.B.
A Structure Memory for Data Flow Computers, S.M. Thesis, EE & CS Department, August 1977; A047-026

TR-187 Long, W.J.
A Program Writer, Ph.D. Dissertation, EE & CS Department, November 1977; A047-595

TR-188 Bryant, R.E.
Simulation of Packet Communication Architecture Computer Systems, S.M. Thesis, EE & CS Department, November 1977; A048-290

TR-189 Ellis, D.J.
Formal Specifications for Packet Communication Systems, Ph.D. Dissertation, EE & CS Department, November 1977; A048-380

TR-190 Moss, E.B.
Abstract Data Types in Stack Based Languages, S.M. Thesis, EE & CS Department, February 1978; A052-332

TR-191 Yonezawa, A.
Specification and Verification Techniques for Parallel Programs Based on Message Passing Semantics, Ph.D. Dissertation, EE & CS Departmert, January 1978; A051-149

TR-192 Niamir, B.
Attribute Partitioning in a Self-Adaptive Relational Data Base System, S.M. Thesis, EE & CS Department, January 1978; A053-292

TR-193 Schaffert, C.
A Formal Definition of CLU, S.M. Thesis, EE & CS Department, January 1978

TR-194 Hewitt, C. and Baker, H.G.
Actors and Continuous Functionals, February 1978; A052-266

TR-195 Bruss, A.R.
On Time-Space Classes and Their Relation to the Theory of Real Addition, S.M. Thesis, EE & CS Department, March 1978

TR-196 Schroeder, M.D., Clark, D.D., Saltzer, J.H. and Wells, D.
Final Report of the Multics Kernel Design Project, March 1978

TR-197 Baker, H.G.
Actor Systems for Real-Time Computation, Ph.D. Dissertation, EE & CS Department, March 1978; A053-328

TR-198 Halstead, R.H.
Multiple-Processor Implementations of Message-Passing Systems, S.M. Thesis, EE & CS Department, April 1978; A054-009

TR-199 Terman, C.J.
The Specification of Code Generation Algorithms, S.M. Thesis, EE & CS Department, April 1978; A054-301

TR-200 Harel, D.
Logics of Programs: Axiomatics and Descriptive Power, Ph.D. Dissertation, EE & CS Department, May 1978

TR-201 Scheifler, R.
A Denotational Semantics of CLU, S.M. Thesis, EE & CS Department, May 1978

TR-202 Principato, R.N.
A Formalization of the State Machine Specification Technique, S.M. & E.E. Thesis, EE & CS Department, July 1978

TR-203 Laventhal, M.S.
Synthesis of Synchronization Code for Data Abstractions, Ph.D. Dissertation, EE & CS Department, July 1978; A058-232

TR-204 Teixeira, T.J.
Real-Time Control Structures for Block Diagram Schemata, S.M. Thesis, EE & CS Department, August 1978; A061-122

TR-205 Reed, D.P.
Naming and Synchronization in a Decentralized Computer System, Ph.D. Dissertation, EE & CS Department, October 1978; A061-407

TR-206 Borkin, S.A.
Equivalence Properties of Semantic Data Models for Data Base Systems, Ph.D. Dissertation, EE & CS Department, January 1979; A066-386

TR-207 Montgomery, W.A.
Robust Concurrency Control for a Distributed Information System, Ph.D. Dissertation, EE & CS Department, January 1979; A066-996

TR-208 Krizan, B.C.
A Minicomputer Network Simulation System, S.B. & S.M. Thesis, EE & CS Department, February 1979

TR-209 Snyder, A.
A Machine Architecture to Support an Object-Oriented Language, Ph.D. Dissertation, EE & CS Department, March 1979; A068-111

TR-210 Papadimitriou, C.H.
Serializability of Concurrent Data Base Updates, March 1979

TR-211 Bloom, T.
Synchronization Mechanisms for Modular Programming Languages, S.M. Thesis, EE & CS Department, April 1979; A069-819

TR-212 Rabin, M.O.
Digitalized Signatures and Public-Key Functions as Intractable as Factorization, January 1979

TR-213 Rabin, M.O.
Probabilistic Algorithms in Finite Fields, January 1979

TR-214 McLeod, D.
A Semantic Data Base Model and Its Associated Structured User Interface, Ph.D. Dissertation, EE & CS Department, March 1979; A068-112

TR-215 Svobodova, L., Liskov, B. and Clark, D.D.
Distributed Computer Systems: Structure and Semantics, April 1979; A070-286

TR-216 Myers, J.M.
Analysis of the SIMPLE Code for Data Flow Computation, May 1979

TR-217 Brown, D.
Storage and Access Costs for Implementations of Variable Length Lists, Ph.D. Dissertation, EE & CS Department, April 1979

TR-218 Ackerman, W.B. and Dennis, J.B.
VAL-A Value-Oriented Algorithmic Language, Preliminary Reference Manual, June 1979; A072-394

TR-219 Sollins, K.R.
Copying Complex Structures in a Distributed System, S.M. Thesis, EE & CS Department, July 1979; A072-441

TR-220 Kosinski, P.R.
Denotational Semantics of Determinate and Nondeterminate Data Flow Programs, Ph.D. Dissertation, EE & CS Department, July 1979

TR-221 Berzins, V.A.
Abstract Model Specifications for Data Abstractions, Ph.D. Dissertation, EE & CS Department, July 1979

TR-222 Halstead, R.H.
Reference Tree Networks: Virtual Machine and Implementation, Ph.D. Dissertation, EE & CS Department, September 1979; A076-570

TR-223 Brown, G.P.
Toward a Computational Theory of Indirect Speech Acts, October 1979; A077-065

TR-224 Isaman, D.L.
Data-Structuring Operations in Concurrent Computations, Ph.D. Dissertation, EE & CS Department, October 1979

TR-225 Liskov, B., Atkinson, R.R., Bloom, T., Moss, E.B., Schaffert, C., Scheifler, R. and Snyder, A.
CLU Reference Manual, October 1979; A077-018

TR-226 Reuveni, A.
The Event Based Language and Its Multiple Processor Implementations, Ph.D. Dissertation, EE & CS Department, January 1980; A081-950

TR-227 Rosenberg, R.L.
Incomprehensible Computer Systems: Knowledge Without Wisdom, S.M. Thesis, EE & CS Department, January 1980

TR-228 Weng, K.-S.
An Abstract Implementation for a Generalized Data Flow Language, Ph.D. Dissertation, EE & CS Department, January 1980

TR-229 Atkinson, R.R.
Automatic Verification of Serializes, Ph.D. Dissertation, EE & CS Department, March 1980; A082-885

TR-230 Baratz, A.E.
The Complexity of the Maximum Network Flow Problem, S.M. Thesis, EE & CS Department, March 1980

TR-231 Jaffe, J.M.
Parallel Computation: Synchronization, Scheduling, and Schemes, Ph.D. Dissertation, EE & CS Department, March 1980

TR-232 Luniewski, A.
The Architecture of an Object Based Personal Computer, Ph.D. Dissertation, EE & CS Department, March 1980; A083-433

TR-233 Kaiser, G.E.
Automatic Extension of an Augmented Transition Network Grammar for Morse Code Conversations, S.B. Thesis, EE & CS Department, April 1980; A084-411

TR-234 Herlihy, M.P.
Transmitting Abstract Values in Messages, S.M. Thesis, EE & CS Department, May 1980; A086-984

TR-235 Levin, L.A.
A Concept of Independence with Applications in Various Fields of Mathematics, May 1980

TR-236 Lloyd, E.L.
Scheduling Task Systems with Resources, Ph.D. Dissertation, EE & CS Department, May 1980

TR-237 Kapur, D.
Towards a Theory for Abstract Data Types, Ph.D. Dissertation. EE & CS Department, June 1980; A085-877

TR-238 Bloniarz, P.A.
The Complexity of Monotone Boolean Functions and an Algorithm for Finding Shortest Paths on a Graph, Ph.D. Dissertation, EE & CS Department, June 1980

TR-239 Baker, C.M.
Artwork Analysis Tools for VLSI Circuits, S.M. & E.E. Thesis, EE & CS Department, June 1980; A087-040

TR-240 Montz, L.B.
Safety and Optimization Transformations for Data Flow Programs, S.M. Thesis, EE & CS Department, July 1980

TR-241 Archer, R.F.
Representation and Analysis of Real-Time Control Structures, S.M. Thesis, EE & CS Department, August 1980; A089-828

TR-242 Loui, M.C.
Simulations Among Multidimensional Turing Machines, Ph.D. Dissertation, EE & CS Department, August 1980

TR-243 Svobodova, L.
Management of Object Histories in the Swallow Repository, August 1980; A089-836

TR-244 Ruth, G.R.
Data Driven Loops, August 1980

TR-245 Church, K.W.
On Memory Limitations in Natural Language Processing, S.M. Thesis, EE & CS Department, September 1980

TR-246 Tiuryn, J.
A Survey of the Logic of Effective Definitions, October 1980

TR-247 Weihl, W.E.
Interprocedural Data Flow Analysis in the Presence of Pointers, Procedure Variables, and Label Variables, S.B. & S.M. Thesis, EE & CS Department, October 1980

TR-248 LaPaugh, A.S.
Algorithms for Integrated Circuit Layout: An Analytic Approach, Ph.D.Dissertation, EE & CS Department, November 1980

TR-249 Turkle, S.
Computers and People: Personal Computation, December 1980

TR-250 Leung, C.K.C.
Fault Tolerance in Packet Communication Computer Architectures, Ph.D. Dissertation, EE & CS Department, December 1980

TR-251 Swartout, W.R.
Producing Explanations and Justifications of Expert Consulting Programs, Ph.D. Dissertation, EE & CS Department, January 1981

TR-252 Arens, G.C.
Recovery of the Swallow Repository, S.M. Thesis, EE & CS Department, January 1981; A096-374

TR-253 Ilson, R.
An Integrated Approach to Formatted Document Production, S.M. Thesis, EE & CS Department, February 1981

TR-254 Ruth, G.R., Alter, S. and Martin, W.A.
A Very High Level Language for Business Data Processing, March 1981

TR-255 Kent, S.
Protecting Externally Supplied Software in Small Computers, Ph.D. Dissertation, EE & CS Department, March 1981

TR-256 Faust, G.G.
Semiautomatic Translation of COBOL into HIBOL, S.M. Thesis, EE & CS Department, April 1981

TR-257 Cesari, C.A.
Application of Data Flow Architecture to Computer Music Synthesis, S.B./S.M. Thesis, EE & CS Department, February 1981

TR-258 Singh, N.P.
A Design Methodology for Self-Timed Systems, S.M. Thesis, EE & CS Department, February 1981

TR-259 Bryant, R.E.
A Switch-Level Simulation Model for Integrated Logic Circuits, Ph.D. Dissertation, EE & CS Department, March 1981

TR-260 Moss, E.B.
Nested Transactions: An Approach to Reliable Distributed Computing, Ph.D. Dissertation, EE & CS Department, April 1981; A100-754

TR-261 Martin, W.A., Church, K.W. and Patil, R.S.
Preliminary Analysis of a Breadth-First Parsing Algorithm: Theoretical and Experimental Results, EE & CS Department, June 1981

TR-262 Todd, K.W.
High Level VAL Constructs in a Static Data Flow Machine, S.M. Thesis, EE & CS Department, June 1981

TR-263 Street, R.S.
Propositional Dynamic Logic of Looping and Converse, Ph.D. Dissertation, EE & CS Department, May 1981

TR-264 Schiffenbauer, R.D.
Interactive Debugging in a Distributed Computational Environment, S.M. Thesis, EE & CS Department, August 1981

TR-265 Thomas, R.E.
A Data Flow Architecture with Improved Asymptotic Performance, Ph.D. Dissertation, EE & CS Department, April 1981

TR-266 Good, M.
An Ease of Use Evaluation of an Integrated Editor and Formatter, S.M. Thesis, EE & CS Department, August 1981

TR-267 Patil, R.S.
Causal Representation of Patient Illness for Electrolyte and Acid-Base Diagnosis, Ph.D. Dissertation, EE & CS Department, October 1981

TR-268 Guttag, J.V., Kapur, D., Musser, D.R.
Derived Pairs, Overlap Closures, and Rewrite Dominoes: New Tools for Analyzing Term Rewriting Systems, EE & CS Department, December 1981

TR-269 Kanellakis, P.C.
The Complexity of Concurrency Control for Distributed Data Bases, Ph.D. Dissertation, EE & CS Department, December 1981

TR-270 Singh, V.
The Design of a Routing Service for Campus-Wide Internet Transport, S.M. Thesis, EE & CS Department, January 1982

TR-271 Rutherford, C.J., Davies, B., Barnett, A.I. and Desforges, J.F.
A Computer System for Decision Analysis in Hodgkins Disease, EE & CS Department, February 1982

TR-272 Smith, B.C.
Reflection and Semantics in a Procedural Language, Ph.D. Dissertation, EE & CS Department, January 1982

TR-273 Estrin, D.L.
Data Communications via Cable Television Networks: Technical and Policy Considerations, S.M. Thesis, EE & CS Department, May 1982

TR-274 Leighton, F.T.
Layouts for the Shuffle-Exchange Graph and Lower Bound Techniques for VLSI, Ph.D. Dissertation, EE & CS Department, August 1982

TR-275 Kunin, J.S.
Analysis and Specification of Office Procedures, Ph.D. Dissertation, EE & CS Department, February 1982

TR-276 Srivas, M.K.
Automatic Synthesis of Implementations for Abstract Data Types from Algebraic Specifications, Ph.D. Dissertation, EE & CS Department, June 1982

TR-277 Johnson, M.G.
Efficient Modeling for Short Channel MOS Circuit Simulation, S.M. Thesis, EE & CS Department, August 1982

TR-278 Rosenstein, L.S.
Display Management in an Integrated Office, S.M. Thesis, EE & CS Department, January 1982

TR-279 Anderson, T.L.
The Design of a Multiprocessor Development System, S.M. Thesis, EE & CS Department, September 1982

TR-280 Guang-Rong, G.
An Implementation Scheme for Array Operations in Static Data Flow Computers, S.M. Thesis, EE & CS Department, May 1982

TR-281 Lynch, N.A.
Multilevel Atomicity - A New Correctness Criterion for Data Base Concurrency Control, EE & CS Department, August 1982

TR-282 Fischer, M.J., Lynch, N.A. and Paterson, M.S.
Impossibility of Distributed Consensus with One Faulty Process, EE & CS Department, September 1982

TR-283 Sherman, H.B.
A Comparative Study of Computer-Aided Clinical Diagnosis, S.M. Thesis, EE & CS Department, January 1981

TR-284 Cosmadakis, S.S.
Translating Updates of Relational Data Base Views, S.M. Thesis, EE & CS Department, February 1983

TR-285 Lynch, N.A.
Concurrency Control for Resilient Nested Transactions, EE & CS Department, February 1983

TR-286 Goree, J.A.
Internal Consistency of a Distributed Transaction System with Orphan Detection, S.M. Thesis, EE & CS Department, January 1983

TR-287 Bui, T.N.
On Bisecting Random Graphs, S.M. Thesis, EE & CS Department, March 1983

TR-288 Landau, S.E.
On Computing Galois Groups and its Application to Solvability by Radicals, Ph.D. Dissertation, EE & CS Department, March 1983

TR-289 Sirbu, M., Schoichet, S.R., Kunin, J.S., Hammer, M.M., Sutherland, J.B. and Zarmer, C.L.
Office Analysis: Methodology and Case Studies, EE & CS Department, March 1983

TR-290 Sutherland, J.B.
An Office Analysis and Diagnosis Methodology, S.M. Thesis, EE & CS Department, March 1983

TR-291 Pinter, R.Y.
The Impact of Layer Assignment Methods on Layout Algorithms for Integrated Circuits, Ph.D. Dissertation, EE & CS Department, August 1983

TR-292 Dornbrook, M. and Blank, M.
The MDL Programming Language Primer, EE & CS Department, June 1980

TR-293 Galley, S.W. and Pfister, G.
The MDL Programming Language, EE & CS Department, May 1979

TR-294 Lebling, P.D.
The MDL Programming Environment, EE & CS Department, May 1980

TR-295 Pitman, K.M.
The Revised Maclisp Manual, EE & CS Department, June 1983

TR-296 Church, K.W.
Phrase-Structure Parsing: A Method for Taking Advantage of Allophonic Constraints, Ph.D. Dissertation, EE & CS Department, June 1983

TR-297 Mok, A.
Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment, Ph.D. Dissertation, EE & CS Department, June 1983; A139-996

TR-298 Krugler, K.
Video Games and Computer Aided Instruction, S.M. Thesis, EE & CS Department, June 1983

TR-299 Wing, J.M.
A Two-Tiered Approach to Specifying Programs, May 1983; A133-949

TR-300 Cooper, G.H.
An Argument for Soft Layering of Protocols, S.M. Thesis, EE & CS Department, August 1983; A133-948

TR-301 Valente, J.A.
Creating a Computer-Based Learning Environment for Physically Handicapped Children, Ph.D. Dissertation, EE & CS Department, September 1983

TR-302 Arvind, Dertouzos, M.L. and Iannucci, R.A.
A Multiprocessor Emulation Facility, October 1983; A136-094

TR-303 Bloom T.
Dynamic Module Replacement in a Distributed Programming System, Ph.D. Dissertation, EE & CS Department, March 1983; A140-619

TR-304 Terman, C.J.
Simulation Tools for Digital LSI Design, Ph.D. Dissertation, EE & CS Department, September 1983; A136-116

TR-305 Bhatt, S.N. and Leighton, F.T.
A Framework for Solving VLSI Graph Layout Problems, October 1983; A136-143

TR-306 Leung, K.C. and Lim, W. Y-P.
PADL - A Packet Architecture Description Language: A Preliminary Reference Manual, October 1983

TR-307 Guttag, J.V. and Horning, J.J.
Preliminary Report on the Larch Shared Language, October 1983; A136-117

TR-308 Oki, B.M.
Reliable Object Storage to Support Atomic Actions, S.M. Thesis, EE & CS Department, May 1983; A136-484

TR-309 Brock, J.D.
A Formal Model of Non-Determinate Dataflow Computation, Ph.D. Dissertation, EE & CS Department, November 1983

TR-310 Granville, R.
Cohesion in Computer Text Generation: Lexical Substitution, S.M. Thesis, EE & CS Department, May 1983; A148-990

TR-311 Burke, G.S., Carrette, G.J. and Eliot, C.R.
NIL Reference Manual, January 1984

TR-312 Lancaster, J.N.
Naming in a Programming Support Environment, S.M. Thesis, EE & CS Department, August 1983; A142-018

TR-313 Koile, K.
The Design and Implementation of an Online Directory Assistance System, S.M. Thesis, EE & CS Department, December 1983; A140-821

TR-314 Weihl, W.E.
Specification and Implementation of Atomic Data Types, Ph.D. Dissertation, EE & CS Department, April 1984; A141823

TR-315 Coan, B.A. and Turpin, R.
Extending Binary Byzantine Agreement to Multivalued Byzantine Agreement, April 1984; A143424

TR-316 Comer, M.H.
Loose Consistency in a Personal Computer Mail System, S.B. & S.M. Thesis, EE & CS Department, May 1984

TR-317 Traub, K.R.
An Abstract Architecture for Parallel Graph Reduction, S.B. Thesis, EE & CS Department, May 1984

TR-318 Ashbell, I.J., M.D.
A Constraint Representation and Explanation Facility for Renal Physiology, S.M. Thesis, EE & CS Department, June 1984

TR-319 Herlihy, M.P.
Replication Methods for Abstract Data Types, Ph.D. Dissertation, EE & CS Department, May 1984; A153-648

TR-320 Kornhauser, D.M.
Coordinating Pebble Motion on Graphs, the Diameter of Permutation Groups, and Applications, S.M. Thesis, EE & CS Department, May 1984

TR-321 Kuszmaul, B.C.
Type Checking in VIM VAL, Ph.D. Dissertation, EE & CS Department, June 1984

TR-322 Moulton, A.S.
Routing the Power and Ground Wires on a VLSI Chip, S.M. Thesis, EE & CS Department, May 1984; A145-356

TR-323 Ackerman, W.B.
Efficient Implementation of Applicative Languages, Ph.D. Dissertation, EE & CS Department, April 1984

TR-324 Schooler, R.
Partial Evaluation as a Means of Language Extensibility, S.M. Thesis, EE & CS Department, August 1984; A148-730

TR-325 Weiss, P.G.
Using Untyped Lambda Calculus to Compute With Atoms, S.M. Thesis, EE & CS Department, February 1984

TR-326 Walker, E.F.
Orphan Detection in the Argus System, S.M. Thesis, EE & CS Department, May 1984; A150-562

TR-327 Chiu, S.Y.
Debugging Distributed Computations in a Nested Atomic Action System, Ph.D. Dissertation, EE & CS Department, December 1984; A153-617

TR-328 Carnese, D.J.
Multiple Inheritance in Contemporary Programming Languages, September 1984

TR-329 Sacks, E.
Qualitative Mathematical Reasoning, November 1984

TR-330 Sarin, S.K.
Interactive On-Line Conferences, Ph.D. Dissertation, EE & CS Department, June 1984; A154-723

TR-331 Sollins, K.R.
Distributed Name Management, Ph.D. Dissertation, EE & CS Department, February 1985; A154-785

TR-332 Culler, D.E.
Resource Management for the Tagged Token Dataflow Architecture, S.M. Thesis, EE & CS Department, January 1985; A154-773

TR-333 Arnold, J.M.
Parallel Simulation of Digital LSI Circuits, S.M. Thesis, EE & CS Department, February 1984; A154-745

TR-334 Zarmer, C.L.
An Approach to Functional Office Automation, S.M. Thesis, EE & CS Department, April 1984

TR-335 Lundelius, J.
Synchronizing Clocks in a Distributed System, S.M. Thesis, EE & CS Department, August 1984

TR-336 Trilling, S.
Some Implications of Complexity Theory on Pseudo-Random Bit Generation, S.M. Thesis, EE & CS Department, January 1985

TR-337 Seiler, L.D.
A Hardware Assisted Methodology for VLSI Design Rule Checking, Ph.D. Dissertation, EE & CS Department, February 1985

TR-338 Koton, P.A.
Towards a Problem Solving System for Molecular Genetics, May 1985

TR-339 Soley, R.M.
Generic Software for Emulating Multiprocessor Architectures, May 1985; A157-662

TR-340 Wellman, M.P.
Reasoning About Preference Models, S.M. Thesis, EE & CS Department, May 1985

TR-341 Boughton, G.A.
Routing Networks for Packet Communication Systems, Ph.D. Dissertation, EE & CS Department, August 1984

TR-342 Stark, E.W.
Foundations of a Theory of Specification for Distributed Systems, Ph.D. Dissertation, EE & CS Department, August 1984

TR-343 Forgaard, R.
A Program for Generating and Analyzing Term Rewriting Systems, S.M. Thesis, EE & CS Department, September 1984

TR-344 Yelick, K.A.
A Generalized Approach to Equational Unification, S.M. Thesis, EE & CS Department, August 1985; A163-112

TR-345 Estrin, D.L.
Access to Inter-Organization Computer Networks, Ph.D. Dissertation, EE & CS Department, August 1985

TR-346 Cosmadakis, S.S.
Equational Theories and Database Constraints, Ph.D. Dissertation, EE & CS Department, August 1985

TR-347 Morecroft, L.E.
A Relative-Motion Microworld, S.M. Thesis, EE & CS Department, September 1985; A161-856

TR-348 Yedwab, L.
On Playing Well in a Sum of Games, S.M. Thesis, EE & CS Department, August 1985

TR-349 Eisenberg, M.A.
Bochser: An Integrated Scheme Programming System, S.M. Thesis, EE & CS Department, August 1985

TR-350 Seliger, R.
Design and Implementation of a Distributed Program for Collaborative Editing, S.M. Thesis, EE & CS Department, September 1985

TR-351 Bhatt, S.N.
The Complexity of Graph Layout and Channel Routing for VLSI, Ph.D. Dissertation, EE & CS Department, February 1984

TR-352 Lucassen, J.M., Gifford, D.K., Berlin, S.T., and Burmaster, D.E.
Boston Community Information System User Manual (Version 6.0), April 1986; A171-426

TR-353 Jagannathan, S.
Data Backup and Recovery in a Computer Architecture for Functional Programming, S.M. Thesis, EE & CS Department, October 1985

TR-354 Stamos, J.W.
Remote Evaluation, Ph.D. Dissertation, EE & CS Department, January 1986; A169-739

TR-355 Guharoy, B.
Data Structure Management in a Data Flow Computer System, S.M. Thesis, EE & CS Department, May 1985

TR-356 Beckerle, M.J.
Logical Structures For Functional Languages, S.M. Thesis, EE & CS Department, February 1986; A169-368

TR-357 Gibson, J.C.
Computation Management in a Single Address Space System, S.M. Thesis, EE & CS Department, January 1986

TR-358 Chaing, C.J.
Primitives for Real-Time Animation in Three Dimensions, S.M. Thesis, EE & CS Department, April 1986

TR-359 Feldmeier, D.C.
A CATV-Based High-Speed Packet-Switching Network Design, S.M. Thesis, EE & CS Department, April 1986; A171666

TR-360 Kunstaetter, R.
Intelligent Physiologic Modeling, S.M. Thesis, EE & CS Department, April 1986

TR-361 Barrington, D.A.
Bounded Width Branching Programs, Ph.D. Dissertation, EE & CS Department, June 1986

TR-362 Kuszmaul, B.C.
Simulating Applicative Architectures on the Connection Machine, S.M. Thesis, EE & CS Department, June 1986

TR-363 Marantz, J.D.
Exploiting Parallelism in VLSI CAD, S.M. Thesis, EE & CS Department, June 1986

TR-364 Lynch, N., Blaustein, B. and Siegel, M.
Correctness Conditions for Highly Available Replicated Databases, June 1986; A171-427

TR-365 Morais, D.R.
ID World: An Environment for the Development of Dataflow Programs Written in ID. May 1986

TR-366 Younis, S.G.
The Clock Distribution System of the Multiprocessor Emulation Facility, S.B. Thesis, EE & CS Department, June 1986

TR-367 Lynch, N.A. and Merritt, M.
Introduction to the Theory of Nested Transactions, July 1986; A171-428

TR-368 Scheifler, R.W. and Gettys, J.
The X Window System, October 1986; A176-476

TR-369 Moses, Y. and Tuttle, M.R.
Programming Simultaneous Actions Using Common Knowledge, February 1987

TR-370 Traub, K.R.
A Compiler for the MIT Tagged-Token Dataflow Architecture, S.M. Thesis, EE & CS Department, August 1986

TR-371 Gao G.R.
A Pipelined Code Mapping Scheme for Static Data Flow Computers, Ph.D. Dissertation. EE & CS Department, August 1986

TR-372 Maley, F.M.
Compaction With Automatic Jog Introduction, S.M. Thesis, EE & CS Department, November 1986; A 176-525

TR-373 Segal, D.A., Gifford, D.K., Lucassen, J.M., Henderson J.B., Berlin, G.T. and Burmaster, D.E.
Boston Community Information System User's Manual (Version 8.17), October 1987; A190-380

TR-374 Goldberg, A.V.
Efficient Graph Algorithms for Sequential and Parallel Computers, Ph.D. Dissertation. EE & CS Department, February 1987; A178-403

TR-375 Osborne, R.B.
Modeling the Performance of the Concert Multiprocessor, S.M. Thesis, EE & CS Department, May 1987; A183-619

TR-376 Day, M.S.
Replication and Reconfiguration in a Distributed Mail Repository, S.M. Thesis, EE & CS Department, May 1987; A186-967

TR-377 Ng, P.
Long Atomic Computations, Ph.D. Dissertation, EE & CS Department, October 1986; A174788

TR-378 Levitin, S.M.
MACE: A Multiprocessing Approach to Circuit Extraction, S.M. Thesis, EE & CS Department, October 1986

TR-379 Sloan, R.H.
The Notion of Security for Probabilistic Public Key Cryptosystems, S.M. Thesis EE & CS Department, October 1986

TR-380 Bradley, E.
Logic Simulation on a Multiprocessor, S.M. Thesis, EE & CS Department, October 1986; A175-776

TR-381 Sherman, A.T.
Cryptology and VLSI (a two-part dissertation), Ph.D. Dissertation, EE & CS Department, October 1986; A175-853

TR-382 Chien, A.A.
Congestion Control in Routing Networks, S.M. Thesis, EE & CS Department October 1986; A175-785

TR-383 Strauss, M.M.
The Organization of Research in the Information Sciences, Case Studies in Japan and in the US, S.M. Thesis, EE & CS Department, December 1986

TR-384 Gifford, D. and Glasser, N.
Remote Pipes and Procedures for Efficient Distributed Communication, October 1986, Revised July 1987

TR-385 Dennis,J.
Data Flow Computer Architecture Final Report, October 1987

TR-386 St. Pierre, M.A.
A Simulation Environment for Schema, S.M. Thesis, EE & CS Department, December 1986

TR-387 Lynch, N.A. and Tuttle, M.
Hierarchical Correctness Proofs for Distributed Algorithms, S.M. Thesis, EE & CS Department, April 1987

TR-388 Hirsch, D.E.
An Expert System for Diagnosing Gait in Cerebral Palsy Patients, S.M. Thesis, EE & CS Department, May 1987

TR-389 Kohane, I.S.
Temporal Reasoning in Medical Expert Systems, Ph.D. Dissertation, EE & CS Department, May 1987

TR-390 Goldman, K.J.
Data Replication in Nested Transaction Systems, S.M. Thesis, EE & CS Department, May 1987; A182-178

TR-391 Goldman, S.
Efficient Methods for Calculating Maximum Entropy Distributions, S.M. Thesis, EE & CS Department, May 1987

TR-392 Kolodney, L.K.
MAM: A Semi-Automatic Debugging Tool for Distributed Programs, S.M. Thesis, EE & CS Department, June 1987

TR-393 Chu, T.-A.
Synthesis of Self-Timed VLSI Circuits from Graph-Theoretic Specifications, Ph.D. Dissertation, EE & CS Department, June 1987

TR-394 Bodlaender, H.L.
Dynamic Programming on Graphs with Bounded Treewidth, June 1987

TR-395 Nuth, P.R.
Communication Patterns in a Symbolic Multiprocessor, S.M. Thesis, EE & CS Department, June 1987: A186-896

TR-396 Jang, Y.
KOLA: Knowledge Organization LAnguage, S.M. Thesis, EE & CS Department, October 1988

TR-397 Gifford, D.K., Heitmann, D., Segal, D.A., Cote, R.G., Tanacea, K. and Burmaster, D.E.
Boston Community Information System 1986 Experimental Test Results, August 1987; AD A187-028

TR398 Gifford, D.K., Cote, R.G. and Segal, D.A.
Clipping Service User's Manual (Version 1.2), September 1987; AD 192-543

TR-399 Gifford,D.K., Cote, R.G. and Segal, D.A.
Walter User's Manual (Version 1.0), September 1987; A192-542

TR-400 Liskov, B., Day, M., Herlihy, M., Johnson, P. and Leavens, G.
Argus Reference Manual, November 1987; A190-382

TR-401 Baldwin, R.
Rule Based Analysis of Computer Security, Ph.D. Dissertation, EE & CS Department, March 1988; A195-736

TR-402 Miller, J.
Multi-Scheme: A Parallel Processing System Base on MIT Scheme, Ph.D. Dissertation, EE & CS Department, September 1987; A190-383

TR-403 Maley, F.M.
Single-Layer Wire Routing, Ph.D. Dissertation, EE & CS Department, August 1987; A186-990

TR-404 Kolodner, E.K.
Recovery Using Virtual Memory, S.M. Thesis, EE & CS Department, July 1987; A187-098

TR-405 Zachary, J.L.
A Framework for Incorporating Abstraction Mechanisms into the Logic Programming Paradigm, Ph.D. Dissertation, EE & CS Department, August 1987; A190-381

TR-406 Muldrow, W.K.
Calvin: A Rule-Based Expert System for Improving Arrhythmia Detector Performance During Noisy ECGs, S.M. Thesis, EE & CS Department, September 1987.

TR-407 Gifford, D., Jouvelot, P., Lucassen, J. and Sheldon, M.
FX-87 Reference Manual, September 1987; A187-031

TR-408 Lucassen, J.
Types and Effects-Towards the Integration of Functional and Imperative Programming, Ph.D. Dissertation, EE & CS Department, August 1987; A186-930

TR-409 Ladin, R., Liskov, B., and Shrira, L.
A Technique for Constructing Highly-Available Services, January 1988; A192-544

TR-410 Jing-Hwa Hwang, D.
Constructing a Highly-Available Location Service for a Distributed Environment, S.M. Thesis, EE & CS Department, January 1988; AD 192-723

TR-411 Kaliski, B.
Elliptic Curves and Cryptography: A Pseudorandom Bit Generator and Other Tools, Ph.D. Dissertation, EE & CS Department, January 1988

TR-412 Berger, B. and Shor, P.W.
Approximation Algorithms for the Maximum Acyclic Subgraph Problem, September 1989

TR-413 Schapire, R.E.
Diversity-Based Inference of Finite Automata, S.M. Thesis, EE & CS Department, May 1988

TR-414 Hicks, J.E.
A High-level Signal Processing Programming Language, S.M. Thesis, EE & CS Department, March 1988

TR-415 Margolus, N.H.
Physics and Computation, Ph.D. Dissertation, Physics Department, March 1988

TR-416 Sacks, E.P.
Automatic Qualitative Analysis of Ordinary Differential Equations Using Piecewise Linear Approximations, Ph.D. Dissertation, EE & CS Department, March 1988

TR-417 Traub, K.R.
Sequential Implementation of Lenient Programming Languages, Ph.D. Dissertation, EE & CS Department, October 1988; A200-984

TR-418 Iannucci, R.A.
A Dataflow/Von Neumann Hybrid Architecture, Ph.D. Dissertation, EE & CS Department, July 1988

TR-419 Jackson, D.
Composing Data & Process Descriptions in the Design of Software Systems, S.M. Thesis, EE & CS Department, May 1988; A198-096

TR-420 Gifford, D.K.
Polychannel Systems for Mass Digital Communication, July 1988

TR-421 Hammel, R.T. and Gifford, D.K.
FX-87 Performance Measurements: Dataflow Implementation, November 1988; A203-150

TR-422 Gifford, D.K. and Segal, D.A.
Boston Community Information System 1987-1988 Experimental Text Results, May 1989

TR-423 Oki, B.M.
Viewstamped Replication for Highly Available Distributed Systems, Ph.D. Dissertation, EE & CS Department, May 1989

TR-424 Xu, A.S.
A Fault-Tolerant Network Kernel for Linda, S.M. Thesis, EE & CS Department, August 1988

TR-425 Maa, G.K.
Code-Mapping Policies for the Tagged-Token Dataflow Architecture, S.M. Thesis, EE & CS Department, May 1988; A198054

TR-426 Klein, P.N.
Efficient Parallel Algorithms for Planar, Chordal, and Interval Graphs, Ph.D. Dissertation, EE & CS Department, October 1988

TR-427 Wellman, M.P.
Formulation of Tradeoffs in Planning Under Uncertainty, Ph.D. Dissertation, EE & CS Department, August 1988

TR-428 Iyengar, A.K.
Parallel DNA Sequence Analysis, S.M. Thesis, October 1988

TR-429 Perlman, R.
Network Layer Protocols with Byzantine Robustness, Ph.D. Dissertation, EE & CS Department, October 1988

TR-430 Plotkin, S.A.
Graph-Theoretic Techniques for Parallel, Distributed, and Sequential Computation, Ph.D. Dissertation, EE & CS Department, September 1988; A200-989

TR-431 Perl, S.E.
Distributed Commit Protocols for Nested Atomic Actions, S.M. Thesis, EE & CS Department, November 1988; A203-900

TR-432 Papadopoulos, G.M.
Implementation of a General Purpose Dataflow Multiprocessor, Ph.D. Dissertation, EE & CS Department, December 1988; A207-609

TR-433 Rosenberg, R.L.
Computer Literacy Education, Ph.D. Dissertation, EE & CS Department, January 1989; AD 209126

TR-434 Jagannathan, S.
A Programming Language Supporting First-Class Parallel Environments, Ph.D. Dissertation, EE & CS Department, January 1989; A207-833

TR-435 Berger, B. and Rompel, J.
Simulating $(log^c n)$-wise Independence in NC, May 1989; A213-974

TR-436 Berger, B.
Data-Structures for Removing Randomness, December 1988

TR-437 Kravets, D.
Finding Farthest Neighbors in a Convex Polygon and Related Problems, S.M. Thesis, EE & CS Department, January 1989; A210-727

TR-438 Heller, S.K.
Efficient Lazy Data-Structures on a Dataflow Machine, February 1989; A209-177

TR-439 Leavens, G.T.
Verifying Object-Oriented Programs that Use Subtypes, February 1989

TR-440 Aiello, W.A.
Proofs, Knowledge and Oracles Three Complexity Results for Interactive Proofs and Zero-Knowledge, Ph.D. Dissertation, EE & CS Department, February 1989

TR-441 Koton, P.A.
Using Experience in Learning and Problem Solving, Ph.D. Dissertation, EE & CS Department, March 1989

TR-443 Soley, R.M.
On the Efficient Exploitation of Speculation Under Dataflow Paradigms of Control, Ph.D. Dissertation, EE & CS Department, May 1989; A214-112

TR-444 Berger, B., Rompel, J. and Shor, P.
Efficient NC Algorithms for Set Cover with Applications to Learning and Geometry, May 1989: AD 213-975

TR-445 Blum, A.
On the Computation Complexity of Training Simple Neural Networks, S.M. Thesis, EE & CS Department, May 1989

TR-446 Culler, D.E.
Managing Parallelism and Resources in Scientific Dataflow Programs, Ph.D. Dissertation, EE & CS Department, March 1989

TR-447 Fortnow, L.J.
Complexity-Theoretic Aspects of Interactive Proof Systems, Ph.D. Dissertation, EE & CS Department, May 1989

TR-448 Sloan, R.H.
Computational Learning Theory: New Models and Algorithms, Ph.D. Dissertation, EE & CS Department, May 1989

TR-449 Onanian, J.S.
A Signal Processing Language for Coarse Grain Dataflow Multiprocessors, S.M. Thesis, EE & CS Department, June 1989; A213 863

TR-450 MIT Computer Science Research Symposium
Copies of Speakers' Viewgraphs, Viewgraphs used for talks given at the Symposium celebrating the 25th Anniversary of Project MAC, October 1988

TR-452 Cherian, M.M.
A Study of Backoff Barrier Synchronization, S.M. Thesis, EE & CS Department, June 1989

TR-453 Gruber, R.E.
Optimistic Concurrency Control for Nested Distributed Transactions, S.M. Thesis, EE & CS Department, June 1989; A213 828

Annual Reports

- Project MAC Progress Report I, to July 1964; AD 465088
- Project MAC Progress Report II, July 1964-July 1965; AD 629494
- Project MAC Progress Report III, July 1965-July 1966; AD 648346
- Project Mac Progress Report IV, July 1966-July 1967; AD 681342
- Project MAC Progress Report V, July 1967-July 1968; AD 687770
- Project MAC Progress Report VI, July 1968-July 1969; AD 705434
- Project MAC Progress Report VII, July 1969-July 1970; AD 732767
- Project MAC Progress Report VIII, July 1970-July 1971; AD 735148
- Project MAC Progress Report IX, July 1971-July 1972; AD 756689
- Project MAC Progress Report X, July 1972-July 1973; AD 771428
- Project MAC Progress Report XI, July 1973-July 1974; A004-966
- Laboratory for Computer Science Progress Report XII, July 1974-July 1975; A024-527
- Laboratory for Computer Science Progress Report XIII, July 1975-July 1976; A061-246
- Laboratory for Computer Science Progress Report XIV, July 1976-July 1977; A061-932
- Laboratory for Computer Science Progress Report 15, July 1977-July 1978; A073-958
- Laboratory for Computer Science Progress Report 16, July 1978-July 1979; A088-355
- Laboratory for Computer Science Progress Report 17, July 1979-July 1980; A093-384
- Laboratory for Computer Science Progress Report 18, July 1980-June 1981; A127586
- Laboratory for Computer Science Progress Report 19, July 1981-June 1982; A143429
- Laboratory for Computer Science Progress Report 20, July 1982-June 1983; A145134
- Laboratory for Computer Science Progress Report 21, July 1983-June 1984; A154810
- Laboratory for Computer Science Progress Report 22, July 1984-June 1985
- Laboratory for Computer Science Progress Report 23, July 1985-June 1986
- Laboratory for Computer Science Progress Report 24, July 1986-June 1987
- Laboratory for Computer Science Progress Report 25, July 1987-June 1988

Copies of all reports with A; AD, or PB numbers listed in Publications may be secured from the National Technical Information Service, U.S. Department of Commerce, Reports Division, 5285 Port Royal Road, Springfield, Virginia 22161 (tel: 703-487-4650). Prices vary. The reference number must be supplied with the request.

References

[1] H. Abelson and G. Sussman. *Structure and Interpretation of Computer Programs.* MIT Press, 1984.

[2] K. Abrahamson. On achieving consensus using a shared memory. In *Proceedings of the Seventh ACM Symposium on Principles of Distributed Computing*, pages 291–302, 1988.

[3] Y. Afek, B. Awerbuch, and H. Moriel. Overhead of resetting a communication protocol is independent of the size of the network. May 1989. Unpublished manuscript.

[4] Y. Afek and E. Gafni. End-to-end communication in unreliable networks. In *Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing*, pages 131–148, Toronto, Ontario, Canada, ACM SIGACT and ACM SIGOPS, 1988.

[5] A. Agarwal. *Trace Compaction Using Cache Filtering with Blocking.* Computer Systems Laboratory, Technical Report TR 88-347, Stanford University, January 1988.

[6] A. Agarwal and M. Cherian. Adaptive backoff synchronization techniques. In *Proceedings of the 16 th Annual International Symposium on Computer Architecture*, IEEE, June 1989.

[7] A. Agarwal and A. Gupta. *Temporal, Processor, and Spatial Locality in Multiprocessor Memory References.* Technical Report, MIT VLSI Memo, 1989. Submitted for publication.

[8] A. Agarwal, M. Horowitz, and J. Hennessy. An analytical cache model. *ACM Transactions on Computer Systems*, May 1989.

[9] A. Agarwal, B. Lim, D. Kranz, and J. Kubiatowicz. April: a processor architecture for multiprocessing. In *17 th Annual International Symposium on Computer Architecture*, Seattle, WA, May 1990.

[10] A. Agarwal, R. Simoni, J. Hennessy, and M. Horowitz. An evaluation of directory schemes for cache coherence. In *Proceedings of the 15 th International Symposium on Computer Architecture*, IEEE, New York, June 1988.

[11] A. Aggarwal, M. Klawe, S. Moran, P. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2:209–233, 1987.

[12] A. Aggarwal and D. Kravets. A linear time algorithm for finding all farthest neighbors in a convex polygon. *Information Processing Letters*, 31:16–20, 1989.

[13] A. Aggarwal and J. Park. Notes on searching in multidimensional monotone arrays In *29 th Symposium on Foundations of Computer Science*, pages 597–512, IEEE, 1988.

[14] A. Aggarwal and J. Park. Sequential searching in multidimensional monotone arrays. 1989. Submitted for publication.

[15] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, November 1987.

[16] A. Appel and T. Jim. Continuation-passing, closure-passing style. In *16 th Symposium on Principles of Programming Languages*, ACM, 1989.

[17] E. Arjomandi, M. Fischer, and N. Lynch. Efficiency of synchronous versus asynchronous distributed systems. *Journal of the ACM*, 30(3):449–456, July 1983.

[18] J. Aspnes, A. Fekete, N. Lynch, M. Merritt, and W. Weihl. A theory of timestamp-based concurrency control for nested transactions. In *Proceedings of the 14 th International Conference on Very Large Data Bases*, pages 431–444, Los Angeles, CA, August 1988.

[19] J. Aspnes and M. Herlihy. Fast randomized consensus using shared memory. Submitted for publication

[20] H. Attiya, A. Bar-Noy, D. Dolev, D. Koller, D. Peleg, and R. Reischuk. Achievable cases in an asynchronous environment. In *Proceedings of 28 th Annual Symposium on Foundations of Computer Science*, pages 337–346, October 1987.

[21] H. Attiya, D. Dolev, and N. Shavit. Bounded polynomial randomized consensus. To appear in *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*, 1989.

[22] H. Attiya, M. Fischer, D. Wang, and L. Zuck. Reliable communication over an unreliable channel. In progress.

[23] H. Attiya and N. Lynch. Time bounds for real-time process control in the presence of timing uncertainty. Submitted for publication.

[24] H. Attiya and M. Tuttle. Bounds for slotted ℓ-exclusion. February 1989. Unpublished manuscript.

[25] B. Awerbuch. On the effects of feedback in dynamic network protocols. In *29 th Annual Symposium on Foundations of Computer Science*, pages 231–245, IEEE, October 1988.

[26] B. Awerbuch. Distributed shortest paths algorithms. In *Proceedings of the 21 st Annual ACM Symposium on Theory of Computing*, Seattle, WA, pages 230–240, ACM SIGACT May 1989.

[27] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Compact distributed data structures for adaptive network routing. In *Proceedings of the 21 st Annual ACM Symposium on Theory of Computing*, Seattle, WA, pages 230–240, ACM SIGACT, ACM, May 1989.

[28] B. Awerbuch, A. Goldberg, M. Luby, and S. Plotkin. Network decomposition and locality in distributed computation. 1989. To appear.

[29] B. Awerbuch, O. Goldreich, and A. Herzberg. A quantitative approach to dynamic networks. May 1989. Unpublished manuscript.

[30] B. Awerbuch, O. Goldreich, D. Peleg, and R. Vainish. *On the Message Complexity of Broadcast: Basic Lower Bound.* Technical Memo TM-365, MIT Laboratory for Computer Science, July 1988. (Accepted for publication in *Journal of the ACM.*).

[31] B. Awerbuch, Y. Mansour, and N. Shavit. Polynomial end-to-end communication. In *30 th Annual Symposium on Foundations of Computer Science*, IEEE, 1989. To appear.

[32] B. Awerbuch and M. Sipser. Dynamic networks are as fast as static networks. In *29 th Annual Symposium on Foundations of Computer Science*, pages 206–220, IEEE, October 1988.

[33] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols and pseudorandom generators for logspace. In *Proceedings of the 21 st STOC Symposium*, ACM, 1989.

[34] P. Barth and R. Nikhii. *Supporting State-sensitive Computation in a Dataflow System.* Computation Structures Group Memo 294, MIT Laboratory for Computer Science, March 1989.

[35] J. M. Barzdin and R. V. Frievald. On the prediction of general recursive functions. *Soviet Mathematics Doklady*, 13:1224–1228, 1972.

[36] D. Beaver and S. Goldwasser. Multi-party computation with faulty majority March 1989. Unpublished.

[37] M. Bellare and S. Goldwasser. New paradigms for digital signature schemes and message authentication based on non-interactive zero knowledge proofs. March 1989. Unpublished.

[38] M. Bellare and S. Micali. Non-interactive oblivious transfer and applications. March 1989. Unpublished.

[39] M. Bellare, S. Micali, and R. Ostrovsky. Parallelizing zero knowledge proofs and perfect completeness zero knowledge. April 1989. Unpublished.

[40] S. Ben-david, G. M. Benedek, and Y. Mansour. The passive student is really weaker. In *COLT*, 1989. Submitted for publication.

[41] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway. Everything provable is provable in zero-knowledge. In *Proceedings of CRYPTO-88*, Springer-Verlag, 1988.

[42] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proof systems, removing intractability assumptions. In *Proceedings of the 20 th STOC*, ACM, 1988.

[43] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Efficient identification schemes using two prover interactive proofs. March 1989. Unpublished.

[44] B. Berger. *Data Structures for Removing Randomness.* Technical Report MIT/LCS/TR-436, MIT Laboratory for Computer Science, December 1988.

[45] B. Berger and J. Rompel. A better performance guarantee for approximate graph coloring. *Algorithmica*, 1988.

[46] B. Berger and J. Rompel. Simulating ($\log^c n$)-wise independence in nc. In *30 th Symposium on Foundations of Computer Science*, IEEE, 1989. To appear. Also appeared as Technical Report MIT/LCS/TR-435.

[47] B. Berger, J. Rompel, and P. Shor. Efficient nc algorithms for set cover with applications to learning and geometry. In *30 th Symposium on Foundations of Computer Science*, IEEE, 1989. To appear. Also appeared as Technical Report MIT/LCS/TR-444.

[48] B. Berger and P. Shor. *Tight bounds for the acyclic subgraph problem.* Technical Report TR-413, MIT Laboratory for Computer Science, June 1989. Submitted for publication.

[49] G. Berry, P. Curien, and J. Levy. Full abstraction for sequential languages: the state of the art. In M. Nivat and J. C. Reynolds, editors, *Algebraic Methods in Semantics*, pages 89–132, Cambridge University Press, 1985.

[50] D. Bertsimas and M. Grigni. On the space-filling curve heuristic for the euclidean traveling salesman problem. *Operations Research Letters*, 1989. To appear.

[51] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced: preliminary report. In *15 th Symposium on Principles of Programming Languages*, pages 229–239, ACM, 1988. Final version in preparation for journal submission.

[52] B. Bloom and A. R. Meyer. Experimenting with process equivalence. January 1989. Twelve page extended abstract, to be submitted.

[53] B. Bloom and A. R. Meyer. A remark on the bisimulation of probabilistic processes. In *Logic at Botic '89, Proceedings*, pages 26–40, Volume 363 of Lecture Notes in Computer Science, Springer-Verlag, July 1989.

[54] B. Bloom and J. G. Riecke. LCF should be lifted. In *Proceedings of AMAST*, pages 133–136, 1989.

[55] A. Blum. *On the Computational Complexity of Training Simple Neural Networks.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised by R. Rivest.

[56] A. Blum. An $\tilde{O}(n^{0.4})$-approximation algorithm for 3-coloring (and improved approximation algorithms for k-coloring). In *Proceedings of the 21 st Annual ACM Symposium on Theory of Computing*, Seattle, WA, May 1989.

[57] A. Blum and R. Rivest. Training a 3-node neural network is NP-Complete. In *Advances in Neural Information Processing Systems 1*, pages 494–501, Morgan Kaufmann, 1988. Also presented at the *1988 Workshop on Computational Learning Theory.*

[58] M. Blum, P. Feldman, and S. Micali. Noninteractive zero-knowledge proofs and their applications. In *Proceedings of the 20 th STOC*, ACM, 1988.

[59] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal of Computing*, 13(4):850–864, 1984.

[60] R. B. Boppana and M. Sipser. The complexity of finite functions. 1989. To appear in *Handbook of Theoretical Computer Science.*

[61] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *JCSS* , 37:156–189, 1988.

[62] G. Brassard and C. Crépeau. Sorting out zero-knowledge. In *Advances in Cryptology: Proceedings of Eurocrypt '89*, Lecture Notes in Computer Science, Springer-Verlag, 1989. To appear.

[63] G. Brassard, C. Crépeau, and J. Robert. All-or-nothing disclosure of secrets (extended abstract). In A. M. Odlyzko, editor, *Advances in Cryptology: Proceedings of CRYPTO '86*, pages 234–238, Volume 263 of Lecture Notes in Computer Science, Springer-Verlag, 1986.

[64] G. Brassard, C. Crépeau, and J. Robert. Information theoretic reductions among disclosure problems. In *27 th Symposium on Foundations of Computer Science*, pages 168–173, IEEE, 1986.

[65] G. Brassard, C. Crépeau, and M. Yung. Everything in NP can be argued in perfect zero-knowledge in a constant number of rounds. In *16 th ICALP*, Lecture Notes in Computer Science, Springer-Verlag, 1989. To appear.

[66] J. Bresnan, editor. *The Mental Representation of Grammatical Relations.* MIT Press, 1982.

[67] M. Bridgeland and R. Watro. Fault tolerant decision making in totally asynchronous distributed systems. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 52–63, August 1987.

[68] J. Brock and W. Ackerman. Scenarios: a model of non determinate computation. In G. Goos and J. Hartmanis, editors, *Formalization of Programming Concepts*, pages 252–259, Volume 107 of Lecture Notes in Computer Science, Springer-Verlag, 1981.

[69] J. Burns and N. Lynch. Mutual exclusion using indivisible reads and writes. In *Proceedings of 18 th Annual Allerton Conference on Communications, Control, and Computing*, pages 833–842, 1980. Submitted for publication.

[70] J. Burns and G. Peterson. *The Ambiguity of Choosing.* Technical Report GIT-ICS-89/12, Georgia Institute of Technology, February 1989.

[71] A. Califano, N. Margolus, and T. Toffoli. *CAM-6 User's Guide (second edition).* Systems Concepts, San Francisco, CA, 1989.

[72] N. Carriero and D. Gelernter. Linda in context. *Communications of the ACM*, 32(4):444–458, 1989.

[73] J. D. Case, M. S. Fedor, M. L. Schoffstall, and J. R. Davin. *A Simple Network Management Protocol.* Request for Comments 1067, DDN Network Information Center, SRI International, August 1988.

[74] J. D. Case, M. S. Fedor, M. L. Schoffstall, and J. R. Davin. *A Simple Network Management Protocol.* Request for Comments 1098, DDN Network Information Center, SRI International, April 1989.

[75] D. Chaiken. Specification for Simulating a Cache Coherence Protocol. MIT Laboratory for Computer Science, January 1989.

[76] M. Cherian. *Scalable Coherent Interface.* Papers in COMPCON, 1989.

[77] B. Chopard. To appear.

[78] B. Chopard. Strings: a cellular-automaton model for moving objects. In *Proceedings of a Workshop on Cellular Automata and Modeling of Complex Physical Systems*, Springer-Verlag, 1989. To appear.

[79] D. Clark, M. Lambert, and L. Zhang. Netblt: a high throughput transport protocol. In *Frontiers in Computer Communications Technology: Proceedings of the ACM-SIGCOMM '87*, pages 353–359, Stowe, VT, August 1987.

[80] D. D. Clark. The Design Philosophy of the DARPA Internet Protocols. In *Proceedings ACM SIGCOMM*, pages 106–114, August 1988. Stanford, CA.

[81] D. D. Clark. *Policy Routing in Internet Protocols.* Request for Comments 1102, DDN Network Information Center, SRI International, May 1989.

[82] D. D. Clark, V. Jacobson, J. Romkey, and H. Salwen. An Analysis of TCP Overhead. *IEEE Communications Magazine*, 27(6):23–29, June 1989.

[83] J. Cohen. *Atomic Stable Storage Across a Network.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[84] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* McGraw-Hill/MIT Press, 1989.

[85] S. Cosmadakis. Computing with recursive types (extended abstract). In *Fourth Symposium on Logic in Computer Science*, IEEE, 1989. To appear.

[86] C. Crépeau. Equivalence between two flavours of oblivious transfers (abstract). In C. Pomerance, editor, *Advances in Cryptology: Proceedings of CRYPTO '87*, pages 350–354, Volume 293 of Lecture Notes in Computer Science, Springer-Verlag, 1987.

[87] C. Crépeau. Verifiable disclosure of secrets and applications. In *Advances in Cryptology: Proceedings of Eurocrypt '89*, Lecture Notes in Computer Science, Springer-Verlag, 1989. To appear.

[88] C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions. In *28 th Symposium on Foundations of Computer Science*, pages 42–52, IEEE, 1988.

[89] C. Crépeau and J. Kilian. Weakening security assumptions and oblivious transfer. In *Advances in Cryptology: Proceedings of CRYPTO '88*, Lecture Notes in Computer Science, Springer-Verlag, 1988. To appear.

[90] P. Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming.* John Wiley and Sons, 1986.

[91] W. J. Dally. *A VLSI Architecture for Concurrent Data Structures.* PhD thesis, California Institute of Technology, 1986.

[92] D. Daniels, A. Spector, and D. Thompson. Distributed logging for transaction processing. In *ACM Special Interest Group on Management of Data 1987 Annual Conference*, pages 82–96, 1987.

[93] A. De Santis, S. Micali, and P. Persiano. Bounded interaction zero-knowledge proofs. 1987. Unpublished.

[94] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Proceedings ACM SIGCOMM*, pages 1–12, September 1989. Austin, TX.

[95] D. Dolev and N. Shavit. Bounded concurrent time-stamp systems are constructible. In *Proceedings of the 21 st Annual ACM Symposium on Theory of Computing.*

[96] J. Doyle and R. S. Patil. *Two Dogmas of Knowledge Representation: Language Restrictions, Taxonomic Classifications, and the Utility of Representation Services.* Technical Report MIT/LCS/TM-387.b, MIT Laboratory for Computer Science, September 1989.

[97] P. Elias. Error-free coding. *IEEE Transactions on Information Theory*, PGIT-4:29–37, 1954. Also *Key Papers in the Development of Coding Theory.* Pages 39–47, E. R. Berlekamp, editor, IEEE Press, 1974.

[98] P. Elias. Zero error capacity under list decoding. *IEEE Transactions on Information Theory*, 34:1070–1074, 1988.

[99] P. Elias. *Error-correcting Codes for List Decoding.* Technical Report MIT/LCS/TM-381, MIT Laboratory for Computer Science, February 1989. Submitted for publication.

[100] M. D. Ernst. *Adequate Models for Recursive Program Schemes.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989. Supervised by A.R. Meyer.

[101] J. F. Knight. Technologies for low latency interconnection switches. In *Proceedings of ACM Symposium on Parallel Algorithms and Architectures*, June 1989.

[102] R. Fagin and J. Halpern. Reasoning about knowledge and probability: preliminary report. In M. Y. Vardi, editor, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 277–293, 1988.

[103] M. S. Fedor, J. R. Davin, M. L. Schoffstall, and J. D. Case. The SNMP and Protocol Design for Network Management. In *Proceedings of the IEEE Systems Design and Networks Conference*, Santa Clara, CA, May 1989.

[104] A. Fekete, N. Lynch, M. Merritt, and W. Weihl. Commutativity-based locking for nested transactions. In *Proceedings of the Third International Workshop on Persistent Object Systems*, pages 113–127, Newcastle, Australia, January 1989. Revised version to appear in *JCSS.*

[105] M. Fischer, N. Lynch, J. Burns, and A. Borodin. Distributed FIFO allocation of identical resources using small shared space. *ACM Transactions on Programming Languages and Systems*, 11(1):90–114, January 1989. Revised version of [107].

[106] M. Fischer, N. Lynch, J. Burns, and A. Borodin. The colored ticket algorithm. Technical Memo MIT/LCS/TM-269, MIT Laboratory for Computer Science, August 1983.

[107] M. Fischer, N. Lynch, J. Burns, and A. Borodin. Distributed FIFO allocation of identical resources using small shared space. Technical Memo MIT/LCS/TM-290, MIT Laboratory for Computer Science, October 1985. To appear in *TOPLAS.* Revised version of [106].

[108] M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. In *Proceedings of Second ACM Symposium on Principles of Database Systems*, pages 1–7, Atlanta, GA, March 1983. Also appears as Technical Report MIT/LCS/TR-282, MIT Laboratory for Computer Science, September 1982. Revised version in *Journal of the ACM*, 32(2):374-382, April 1985.

[109] M. Fischer and L. Zuck. *Reasoning About Uncertainty in Fault-tolerant Distributed Systems.* Technical Report YALEU/DCS/TR-643, Yale University, August 1988.

[110] M. J. Fischer, N. A. Lynch, J. E. Burns, and A. Borodin. Resource allocation with immunity to limited process failure. In *20 th Annual Symposium on Foundations of Computer Science*, San Juan, Puerto Rico, pages 234–254, October 1979.

[111] L. Fortnow. *Complexity-theoretic Aspects of Interactive Proof Systems.* PhD thesis, MIT, 1989.

[112] L. Fortnow and M. Sipser. Probabilistic computation and linear time. In *21 st Symposium on Theory of Computing*, ACM, 1989. To appear.

[113] M. Foster and R. I. Greenberg. Lower bounds on the area of finite-state machines. *Information Processing Letters*, 30(1):1–7, January 1989.

[114] M. Fredman and J. Komlos. On the size of separating systems and perfect hash functions. *SIAM J. Algeb. Discr. Meth.*, 5:61–68, 1984.

[115] J. Fried. Broadband module design: cost/performance tradeoffs. Lecture given at International Workshop on Physical Design of Broadband Switching and Multiplexing Equipment, April 1989.

[116] J. Fried. A VLSI chip set for burst and ATM switching. In *International Communications Conference*, 1989.

[117] J. Fried. *VLSI Processor Design for Communications Networks.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised by C.E. Leiserson. Also appears as an MIT VLSI memo.

[118] J. Fried, D. Ghosh, and J. Daly. A novel content-addressable memory circuit. *Electronics Letters*, 1989.

[119] J. Fried and P. Kubat. Reliability models for facilities switching. Submitted to *IEEE Transactions on Reliability*, 1989.

[120] J. Fried and B. Kuszmaul. NAP (no ALU processor): the great communicator. In *Frontiers of Massively Parallel Computation*, 1988. An extended version of this paper has been submitted for publication in the *Journal of Parallel and Distributed Computing.*

[121] U. Frisch, et al. Lattice gas hydrodynamics in two and three dimensions. *Complex Systems*, 1(4):633–707, 1987.

[122] R. Gallager, P. Humblet, and P. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems*, 5(1):66–77, January 1983.

[123] J. R. Glass. *Finding Acoustic Regularities in Speech: Applications to Phonetic Recognition.* PhD thesis, May 1988.

[124] A. V. Goldberg, S. Plotkin, D. B. Shmoys, and E. Tardos. Interior-point methods in parallel computation. 1989. Submitted for publication.

[125] A. V. Goldberg, E. Tardos, and R. E. Tarjan. Network flow algorithms. In *Flows, Paths and VLSI-layout*, Springer Verlag, 1989. To appear.

[126] K. Goldman. Highly concurrent logically synchronous multicast. In *Proceedings of the Third International Workshop on Distributed Algorithms*, September 1989. Longer version as Technical Memo MIT/LCS/TM-401, MIT Laboratory for Computer Science, July 1989. Submitted to *Distributed Computing.*

[127] K. Goldman. *Paralation Views: Abstractions for Efficient Scientific Computing on the Connection Machine.* Technical Memo MIT/LCS/TM-398, MIT Laboratory for Computer Science, August 1989.

[128] S. A. Goldman. A space efficient greedy triangulation algorithm. *Information Processing Letters*, 1989. To appear. Earlier version available as MIT/LCS/TM-366, MIT Laboratory for Computer Science.

[129] S. A. Goldman and R. L. Rivest. Mistake bounds and efficient halving algorithms. 1989. Submitted for publication.

[130] S. A. Goldman, R. L. Rivest, and R. E. Schapire. Learning binary relations and total orders. 1989. To appear.

[131] O. Goldreich, A. Herzberg, and Y. Mansour. Source to destination communication in the presence of faults. In *Eighth Annual ACM Symposium on Principles of Distributed Computing*, 1989. To appear.

[132] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *21 st Symposium on Theory of Computing*, pages 25–32, ACM, 1989.

[133] O. Goldreich, Y. Mansour, and M. Sipser. Interactive proof systems: provers that never fail and random selection. In *Proceedings of the 27 th FOCS*, pages 449–462, IEEE, 1987.

[134] S. Goldschmidt. Simulating multiprocessor address traces. November 1987. Computer Systems Laboratory, Stanford University.

[135] S. Goldwasser and M. Sipser. Arthur-Merlin games verses interactive proof systems. In *Proceedings of the 18 th STOC*, ACM, 1986.

[136] R. I. Greenberg. *Area-universal Networks.* VLSI Memo 524, MIT, 1989.

[137] R. I. Greenberg, A. T. Ishii, and A. L. Sangiovanni-Vincentelli. MulCh: A multilayer channel router using one, two, and three layer partitions. In *IEEE International Conference on Computer-Aided Design,* pages 88–91, IEEE Computer Society Press, 1988.

[138] S. D. Greenwald, R. S. Patil, and R. G. Mark. Improved arrhythmia detection in noisy ECGs through the use of expert systems. In *Computers in Cardiology*, 1988.

[139] M. Grigni and D. Peleg. *Tight Bounds on Minimum Broadcast Networks.* Technical Memo TM-374, MIT Laboratory for Computer Science, December 1988.

[140] R. Gruber. *Optimistic Concurrency Control for Nested Distributed Transactions.* Technical Report MIT/LCS/TR-453, MIT Laboratory for Computer Science, June 1989.

[141] I. J. Haimowitz, R. S. Patil, and P. Szolovits. Representing medical knowledge in a terminological language is difficult. In *Symposium on Computer Applications in Medical Care*, pages 101–105, 1988.

[142] L. A. Hall and D. B. Shmoys. Approximation schemes for constrained scheduling problems. 1989. Submitted for publication.

[143] L. A. Hall and D. B. Shmoys. Jackson's rule: making a good heuristic better. *Mathematics of Operations Research*, 1989. To appear.

[144] J. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*, pages 50–61, 1984. To appear in *JACM.* A revised version appears as *IBM Research Report RJ 4421*, Third Revision, September, 1988.

[145] J. Halpern, Y. Moses, and M. Tuttle. A knowledge-based analysis of zero knowledge. In *Proceedings of the 20 th Annual ACM Symposium on Theory of Computing*, pages 132–147, May 1988.

[146] J. Halpern and M. Tuttle. Knowledge, probability, and adversaries. In *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*, pages 103–118, August 1989. Also, IBM Research Report RJ 7045, September 1989.

[147] R. Halstead. Multilisp: a language for concurrent symbolic computation. *ACM Transactions on Programming Languages and Systems*, 7(4):501–538, October 1985.

[148] R. Halstead. An assessment of Multilisp: lessons from experience. *International Journal of Parallel Programming*, 15(6):459–501, December 1986.

[149] R. Halstead. Parallel symbolic computing. *IEEE Computer*, 19(8):35–43, August 1986.

[150] R. Halstead and T. Fujita. Masa: a multithreaded processor architecture for parallel symbolic computing. In *15 th International Symposium on Computer Architecture*, pages 443–451, Honolulu, HI, June 1988.

[151] R. H. Halstead. Multilisp: A language for parallel symbolic computation. *ACM Transactions on Programming Languages and Systems*, 7(4):501–539, October 1985.

[152] T. Halstead, R., R. Anderson, Osborne, and T. Sterling. Concert: design of a multiprocessor development system. In *13 th International Symposium on Computer Architecture*, pages 40–48, Tokyo, Japan, June 1986.

[153] M. D. Hansen. Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems. 1989. Submitted to *FOCS '89.*

[154] N. Harris. Probabilistic reasoning in the domain of genetic counseling. In *Symposium on Computer Applications in Medical Care*, 1989. Finalist, student paper competition, SCAMC 1989.

[155] N. Harris. *Probabilistic Reasoning in the Domain of Genetic Counseling.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[156] J. Hastad, T. Leighton, and M. Newman. Fast computation using faulty hypercubes. In *Proceedings of the 21 st Annual ACM Symposium on Theory of Computing*, Seattle, WA, ACM SIGACT, May 1989. To appear.

[157] M. Held and R. M. Karp. The traveling salesman problems and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.

[158] S. Heller. *Efficient Lazy Data-structures on a Dataflow Machine.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, January 1989.

[159] D. Helmbold, R. Sloan, and M. Warmuth. Learning nested differences of intersections closed concept classes. 1989. Submitted to *COLT '89.*

[160] M. Herlihy, N. Lynch, M. Merritt, and W. Weihl. On the correctness of orphan elimination algorithms. In *17th IEEE Symposium on Fault-tolerant Computing*, pages 8–13, 1987. Also appears as Technical Memo MIT/LCS/TM-329, MIT Laboratory for Computer Science, May 1987. To appear in *Journal of the ACM.*

[161] A. T. Heybey. *Rate-based Congestion Control in Networks with Smart Links.* Bachelor's thesis, MIT, May 1988.

[162] M. Hill, et al. Design decisions in spur. *IEEE Computer*, 19(10):8–22, November 1986.

[163] M. Horowitz, J. Hennessy, P. Chow, G. Gulak, J. Acken, A. Agarwal, C. Chu, S. McFarling, S. Przybylski, S. Richardson, A. Salz, R. Simoni, D. Stark, P. Steenkiste, S. Tjiang, and M. Wing. A 32-bit microprocessor with 2K-byte on-chip cache. In *Proceedings of the IEEE International Solid-state Circuits Conference*, 1987.

[164] T. Hoshino. *PAX Computer. High-Speed Parallel Processing and Scientific Computing.* Addison-Wesley, 1989. H.S. Stone, editor.

[165] M. Huffman. *A Spatial Locality Based Trace Compaction Method.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, April 1989.

[166] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random number generation from any one-way function. In *21 st Symposium Theory of Computing*, pages 12–24, ACM, 1989.

[167] R. Impagliazzo and M. Yung. Direct minimum knowledge computations. In Advances in Cryptology, 1987.

[168] A. T. Ishii. *A Digital Model for Level-clocked Circuitry.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1988. Supervised by C.E. Leiserson.

[169] S. Jagannathan. *A Programming Language Supporting First-class Parallel Environments.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, January 1989.

[170] L. Jategaonkar and J. C. Mitchell. ML with extended pattern matching and subtypes (preliminary version). In *Symposium on Lisp and Functional Programming*, pages 198–211, ACM, 1988.

[171] L. A. Jategaonkar. *ML With Extended Pattern Matching and Subtypes.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised by A. Meyer. To appear in September 1989.

[172] D. Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3):404–425, July 1985.

[173] G. Kahn. The semantics of a simple language for parallel programming. In J. L. Rosenfeld, editor, *Information Processing 74*, pages 471–475, North-Holland, 1974.

[174] R. Karp, E. Upfal, and A. Widgerson. Constructing a perfect matching is in random NC. *Combinatorica*, 6(1):35–48, 1986.

[175] V. Kathail. *Optimal Interpreters for λ-calculus Based Functional Languages.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected September 1989.

[176] J. Kilian. *Randomness in Algorithms and Protocols.* PhD thesis, MIT Department of Mathematics, 1989. Supervised by S. Goldwasser.

[177] J. Kilian. Efficient zero-knowledge proof systems with bounded interaction. Submitted to FOCS '89.

[178] J. Kilian, S. Kipnis, and C. E. Leiserson. The organization of permutation architectures with bussed interconnections. *IEEE Transactions on Computers*, 1989. To appear. Also appeared as Technical Memo MIT/LCS/TM-379 and VLSI Memo 89-500. Earlier version appeared in *28 th IEEE Annual Symposium on Foundations of Computer Science*, pages 305–315, 1987

[179] J. Kilian and N. Nisan. Space bounded cryptography. Submitted to FOCS '89.

[180] S. Kipnis. *Three Methods for Range Queries in Computational Geometry.* Technical Memo TM-388, MIT Laboratory for Computer Science, March 1989.

[181] P. Klein and C. Stein. *A Parallel Algorithm for Eliminating Cycles in Undirected Graphs.* Center for Research in Computing Technology Technical Report TR-01-89, Harvard University, March 1989. Submitted to *Information Processing Letters.*

[182] R. Koch. Increasing the size of a network by a constant factor can increase performance by more than a constant factor. In *29 th th FOCS*, pages 221–230, IEEE, October 1988.

[183] R. Koch. *An Analysis of the Performance of Interconnection Networks for Multiprocessor Systems.* PhD thesis, MIT Department of Mathematics, 1989. Supervised by F.T. Leighton.

[184] R. Koch, T. Leighton, B. Maggs, S. Rao, and A. Rosenberg. Work-preserving emulations of fixed-connection networks. In *Proceedings of the 21 st Annual ACM Symposium on Theory of Computing*, Seattle, WA, May 1989. To appear.

[185] J. Körner and K. Marton. New bounds for perfect hashing via information theory. *European J. Combinatorics*, 9:523–530, 1986.

[186] W. Kornfeld and C. Hewitt. The scientific community metaphor. *IEEE Transactions on Systems, Man, and Cybernetics*, January 1981.

[187] J. Kowalik. *Parallel MIMD Computation: HEP Supercomputer and Its Applications.* MIT Press, 1985.

[188] D. Kranz. *ORBIT: An Optimizing Compiler for Scheme.* Phd thesis, Technical Report YALEU/DCS/RR-632, Yale University, February 1988.

[189] D. Kranz, R. Halstead, and E. Mohr. Mul-T: A high-performance parallel Lisp. In *Proceedings of SIGPLAN '89, Symposium on Programming Languages Design and Implementation*, pages 81–90, June 1989.

[190] D. Kranz, R. Kelsey, J. Rees, P. Hudak, J. Philbin, and N. Adams. Orbit: an optimizing compiler for Scheme. In *Proceedings of SIGPLAN '86 Symposium on Compiler Construction*, pages 219–233, June 1986.

[191] D. Kravets. *Finding Farthest Neighbors in a Convex Polygon and Related Problems.* Technical Report MIT/LCS/TR-437, MIT Laboratory for Computer Science, January 1989.

[192] R. Ladin. *A Method for Constructing Highly Available Services and a Technique for Distributed Garbage Collection.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[193] R. Ladin, B. Liskov, and L. Shrira. A technique for constructing highly-available services. *Algorithmica,* 3:393–420, 1988.

[194] L. Lamport and N. Lynch. Distributed computing. Chapter of *Handbook on Theoretical Computer Science.* Also, to be published as Technical Memo MIT/LCS/TM-384, MIT Laboratory for Computer Science, 1989.

[195] B. Lampson. Atomic transactions. In *Distributed Systems: Architecture and Implementation,* Lecture Notes in Computer Science, Springer-Verlag, 1981.

[196] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, B50, 1988.

[197] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Sequencing and scheduling: algorithms and complexity. In S. C. Graves, A. H. G. Rinnooy Kan, and P. Zipkin, editors, *The Handbooks of Operations Research and Management Science, Volume IV: Production Planning and Inventory*, North-Holland, 1989. To appear.

[198] D. H. Lawrie. Access and alignment of data in an array processor. *IEEE Transactions on Computers*, C-24(12):1145–1155, December 1975.

[199] G. Leavens. *Verifying Object-oriented Programs that Use Subtypes.* Technical Report MIT/LCS-TR-439, MIT Laboratory for Computer Science, February 1989.

[200] K. Lee. *Automatic Speech Recognition: The Development of the Sphinx System Appendix I.* Kluwer Academic Publishers, 1989.

[201] T. Leighton, M. Newman, A. G. Ranade, and E. Schwabe. Dynamic tree embeddings in butterflies and hypercubes. In *First Symposium on Parallel Algorithms and Architectures*, ACM, 1989.

[202] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *29 th Symposium on Foundations of Computer Science*, pages 422–431, IEEE, 1988.

[203] C. Leiserson. Fat-trees: universal networks for hardware efficient supercomputing. *IEEE Transactions on Computers*, C-34(10), October 1985.

[204] C. E. Leiserson and J. B. Saxe. *Retiming Synchronous Circuitry.* Technical Memo TM-372, MIT Laboratory for Computer Science, October 1988.

[205] L. Levin. Homogeneous measures and polynomial time invariants. In *29 th Symposium on Foundations of Computer Science*, pages 36–41, IEEE, 1988.

[206] L. Levin and R. Venkatesan. Random instances of a graph coloring problem are hard. In *20 th Symposium Theory of Computing*, pages 217–222, ACM, 1988.

[207] B. Lim. Instruction Set Architecture of the APRIL Processor. April 1989. To appear.

[208] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform and learnability. 1989. Submitted for publication.

[209] N. Linial, Y. Mansour, and R. L. Rivest. Results on learnability and the Vapnik-Chervonenkis dimension. In *Proceedings of the 29 th Annual Symposium on Foundations of Computer Science*, pages 120–129, October 1988.

[210] N. Linial and N. Nisan. Approximate inclusion-exclusion. 1989. Submitted for publication.

[211] R. Lipton and R. E. Tarjan. A separator theorem for planar graphs. In *A Conference on Theoretical Computer Science*, University of Waterloo, August 1977.

[212] B. Liskov, T. Bloom, D. Gifford, R. Scheifler, and W. Weihl. Communication in the Mercury system. In *Proceedings of the 21 st Annual Hawaii Conference on System Sciences*, pages 178–187, IEEE, January 1988.

[213] B. Liskov and L. Shrira. Promises: linguistic support for efficient asynchronous procedure calls in distributed systems. In *Proceedings of the ACM SIGPLAN '88 Conference on Programming Languages Design and Implementation*, June 1988.

[214] B. Liskov, L. Shrira, and J. Wroclawski. Efficient at-most-once messages based on synchronized clocks. Submitted for publication.

[215] N. Littlestone. Learning when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[216] W. J. Long, S. Naimi, M. Criscitiello, and G. Larsen. Differential diagnosis generation from a causal network with probabilities. In *Proceedings Computers in Cardiology Conference*, IEEE, 1988.

[217] M. Loui and H. Abu-Amara. Memory requirements for agreement among unreliable asynchronous processes. *Advances in Computing Research*, 4:163–183, 1987.

[218] N. Lynch. Modelling real-time systems. November 1988.

[219] N. Lynch. A hundred impossibility proofs for distributed computing. In *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*, Edmonton, Alberta, Canada, August 1989. Also, Technical Memo MIT/LCS/TR-394, MIT Laboratory for Computer Science, 1989.

[220] N. Lynch and H. Attiya. Assertional proofs of timing properties. In progress.

[221] N. Lynch and K. Goldman. *Distributed Algorithms.* MIT/LCS/RSS-5, MIT Laboratory for Computer Science, 1989. Lecture notes for 6.852.

[222] N. Lynch, Y. Mansour, and A. Fekete. The data link layer: two impossibility results. In *Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing*, pages 149–170, Toronto, Canada, August 1988. Also, Technical Memo MIT/LCS/TM-355, MIT Laboratory for Computer Science, May 1988. Revised Technical Memo MIT/LCS/TM-355.b.

[223] N. Lynch, M. Merritt, W. Weihl, and A. Fekete. *A Theory of Atomic Transactions.* Technical Memo MIT/LCS/TM-362, MIT Laboratory for Computer Science, June 1988. Also in *Second International Conference on Database Theory*, pages 41–71, Bruges, Belgium, September 1988.

[224] N. Lynch, M. Merritt, W. Weihl, and A. Fekete. *Atomic transactions.* Book in progress.

[225] N. Lynch and M. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 137–151, August 1987.

[226] N. Lynch and M. Tuttle. An introduction to input/output automata. To be published in *Centrum voor Wiskunde en Informatica Quarterly*. Also Technical Memo MIT/LCS/TM-373, Laboratory for Computer Science, November 1988.

[227] N. A. Lynch. Concurrency control for resilient nested transactions. *Advances in Computing Research*, 3:335–373, 1986.

[228] N. A. Lynch and E. W. Stark. *A Proof of the Khan Principle for Input/Output Automata.* Technical Memo TM-349, MIT Laboratory for Computer Science, January 1988.

[229] Y. Mansour and B. Schieber. The intractability of bounded protocols for non-fifo channels. In *Eighth Annual ACM Symposium on Principles of Distributed Computing*, 1989. To appear.

[230] Y. Mansour, B. Schieber, and P. Tiwari. The complexity of approximating the square root. In *30 th Annual Symposium on Foundations of Computer Science*, 1989. To appear.

[231] Y. Mansour and B. S. P. Tiwari. Lower bounds for computations with the floor operation. In *Proceedings of ICALP*, 1989. To appear.

[232] N. Margolus and T. Toffoli. Cellular automata machines. *Complex Systems*, pages 967–993, 1987.

[233] A. R. Meyer. *Semantical Paradigms: Notes for an Invited Lecture, with Two appendices by S. Cosmodakis.* Technical Report MIT/LCS/TM-353, MIT Laboratory for Computer Science, July 1988. Also in *Third Symposium Logic in Computer Science*, pages 236–253, IEEE, 1988.

[234] A. R. Meyer and J. G. Riecke. Continuations may be unreasonable. In *Proceedings of Conference on Lisp and Functional Programming*, pages 63–71, ACM, 1988.

[235] A. R. Meyer and M. A. Taitslin, editors. *Logic at Botic,'89: Proceedings of a Symposium on Logical Foundations of Computer Science.* Volume 363 of Lecture Notes in Computer Science, Springer-Verlag, July 1989.

[236] S. Micali and R. Ostrovsky. Simple non-interactive zero knowledge proofs with preprocessing oblivious transfer. September 1988. To appear.

[237] S. Micali and A. Shamir. An improvement of the Fiat-Shamir identification. In *Proceedings of CRYPTO-88*, Springer-Verlag, 1988.

[238] F. Modugno, M. Merritt, and M. Tuttle. Time constrained automata. November 1988. Unpublished manuscript.

[239] J. Mosely. *Asynchronous Distributed Flow Control Algorithms.* Technical Report LIDS-TR-1415, MIT Laboratory for Information and Decision Systems, October 1984.

[240] Y. Moses and M. Tuttle. Programming simultaneous actions using common knowledge. In *Proceedings of the 27 th Annual Symposium on Foundations of Computer Science*, pages 208–221, Toronto, Ontario, Canada, October 1986. Expanded version Technical Report MIT/LCS/TR-369, MIT Laboratory for Computer Science, February 1987. Revised version in *Algorithmica*, 3(1):124-169, 1988.

[241] Y. Moses and M. Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3(1):121–169, 1988. Expanded version of [240].

[242] K. Mulmuley, U. Vazarani, and V. Vazarani. Matching is as easy as matrix inversion. In *19 th Symposium of Theory of Computing*, pages 345–354, ACM, 1987.

[243] R. Nikhil. *Can Dataflow Subsume von Neumann Computing?* Computation Structures Group Memo 292, MIT Laboratory for Computer Science, November 1988.

[244] R. S. Nikhil. *Id (Version 88.1) Reference Manual.* Technical Report, Computation Structures Group Memo 284, MIT Laboratory for Computer Science, August 1988.

[245] N. Nisan. Crew prams and decision trees. In *Proceedings of the 21 st STOC Symposium*, ACM, 1989.

[246] B. M. Oki. *Viewstamped Replication for Highly Available Distributed Systems.* Technical Report MIT/LCS/TR-423, MIT Laboratory for Computer Science, August 1988.

[247] J. B. Orlin. *Genuinely Polynomial Simplex and Non-simplex Algorithms for the Minimum Cost Flow Problem.* Technical Report No. 1615-84, MIT Sloan School of Management, 1984.

[248] J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. In *Proceedings of the 20 th STOC Symposium*, pages 377–387, ACM, 1988.

[249] R. Osborne. *Speculative Computation in Multilisp.* Technical Report MIT/LCS/TR-464, MIT Laboratory for Computer Science, December 1989.

[250] R. Ostrovsky. Noninteractive zero-knowledge proofs with pre-processing oblivious transfer. 1988. Unpublished.

[251] S. Owicki and A. Agarwal. Evaluating the performance of software cache coherence. In *Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS III)*, April 1989.

[252] D. Pallett. Benchmark tests for DARPA resource management database performance evaluations. In *Proceedings ICASSP-89*, May 1989.

[253] G. Papadopoulos. *The Monsoon Processing Element Architecture Reference.* Computation Structures Group Memo 283, MIT Laboratory for Computer Science, March 1988.

[254] J. Park. *Notes on Searching in Multidimensional Monotone Arrays.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, January 1989. Supervised by C. E. Leiserson.

[255] S. G. Pauker and S. P. Pauker. Prescriptive models to support decision making in genetics. In G. Evers-Kiebooms, J. J. Cassiman, H. Vandenbeghe, and G. d'Ydewalle, editors, *Genetic Risk, Risk Perception, and Decision Making*, pages 279–296, Alan R. Liss, 1987.

[256] S. P. Pauker and S. G. Pauker. The amniocentesis decision: ten years of decision analytic experience. In G. Evers-Kiebooms, J. J. Cassiman, H. Vandenbeghe, and G. d'Ydewalle, editors, *Genetic Risk, Risk Perception, and Decision Making*, pages 151–169, Alan R. Liss, 1987.

[257] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[258] R. Perlman. *Network Layer Protocols With Byzantine Robustness.* PhD thesis, MIT, August 1988.

[259] N. Perugini. *Neural Network Learning: Effects of Network and Training Set Size.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

[260] G. Peterson and M. Fischer. Economical solutions for the critical section problem in a distributed system. In *Proceedings of Ninth STOC*, pages 91–97, May 1977.

[261] G. F. Pfister and V. A. Norton. 'Hotspot' contention and combining in multistage interconnection networks. *IEEE Transactions on Computers*, C-34(10), October 1985.

[262] C. Phillips. Parallel graph contraction. In *First Symposium on Parallel Algorithms and Architectures*, ACM, 1989. To appear.

[263] C. Phillips and S. A. Zenios. Experiences with large scale network optimization on the connection machine. In *Impact of Recent Computer Advances on Operations Research*, Elsevier Science Publishing Co., 1989.

[264] M. Phillips. Automatic discovery of acoustic measurements for phonetic classification. *Journal of the Acoustical Society of America*, 84(S216), 1988.

[265] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–257, 1977.

[266] S. Plotkin and E. Tardos. Improved dual network simplex. 1989. Submitted for publication.

[267] M. W. Pozen, R. B. D'Agostino, H. P. Selker, P. A. Sytkowski, and W. B. Hood. A predictive instrument to improve coronary-care-unit admission practices in acute ischemic heart disease. *New England Journal of Medicine*, 310:1273–1278, 1984.

[268] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[269] A. M. Rabinovich. *Nets and Processes.* PhD thesis, Tel Aviv University, 1988. Supervised by B. A. Trakhtenbrot.

[270] J. Rees and N. Adams. T: A dialect of Lisp. In *Proceedings of Symposium on Lisp and Functional Programming*, August 1982.

[271] J. Rees, N. Adams, and J. Meehan. *The T Manual, Fourth Edition.* Technical Report, Yale University, Computer Science Department, January 1984.

[272] J. Rees and W. Clinger. Revised^3 report on the algorithmic language scheme. *ACM SIGPLAN Notices*, 21(12):37–79, December 1986.

[273] J. G. Riecke. *Should a Function Continue?* Master's thesis, MIT Department of Electrical Engineering and Computer Science, January 1989. Supervised by A.R. Meyer.

[274] R. L. Rivest and R. E. Schapire. Inference of finite automata using homing sequences. In *Proceedings of the 21 st Annual ACM Symposium on Theory of Computing*, Seattle, WA, May 1989. To appear.

[275] R. L. Rivest and R. Sloan. Learning complicated concepts reliably and usefully. In *Proceedings AAAI-88*, pages 635–640, American Association for Artificial Intelligence, August 1988.

[276] R. L. Rivest and R. Sloan. A new model for inductive inference. In M. Vardi, editor, *Proceedings of the Second Annual Theoretical Aspects of Reasoning about Knowledge Conference*, pages 13–27, Morgan Kaufmann, March 1988. Submitted to *Information and Computation.*

[277] J. Rompel. *A Better Performance Guarantee for Approximate Graph Coloring.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised by T. Leighton.

[278] T. A. Russ. Using hindsight in medical decision making. *Symposium on Computer Applications in Medical Care*, 1989. Finalist in student paper competition.

[279] A. Salz. VTRACE. Computer Systems Laboratory, Stanford University, 1984.

[280] R. Schaffer. On the Correctness of Atomic Multi-writer Registers. Bachelor's Thesis, MIT, June 1988. Also Technical Memo MIT/LCS/TM-364, MIT Laboratory for Computer Science, 1988.

[281] R. E. Schapire. The strength of weak learnability. 1989. Submitted for publication.

[282] M. L. Schoffstall, J. R. Davin, M. S. Fedor, and J. D. Case. *SNMP Over Ethernet.* Request for Comments 1089, DDN Network Information Center, SRI International, February 1989.

[283] S. Seneff. A joint synchrony/mean-rate model of auditory speech processing. In *Journal of Phonetics*, 16:55–76, January 1988.

[284] S. Seneff. Tina: a probabilistic syntactic parser for speech understanding systems. In *Proceedings from Speech and Natural Language Workshop*, pages 168–178, Philadelphia, PA, 1989.

[285] A. T. Sherman. *VLSI Placement and Routing: The PI Project.* Springer-Verlag, 1989.

[286] D. B. Shmoys and E. Tardos. Computational complexity. In R. Graham, M. Grötschel, and L. Lovász, editors, *The Handbook of Combinatorics*, North-Holland, 1989. To appear.

[287] D. B. Shmoys and D. Williamson. Analyzing the Held-Karp TSP bound: a monotonicity property with application. 1989. Submitted to *Information Processing Letters.*

[288] R. Sloan. Types of noise in data for concept learning. In *First Workshop on Computational Learning Theory*, pages 91–96, Morgan Kaufmann, 1988.

[289] R. Sloan. *All Zero-knowledge Proofs are Proofs of Language Membership.* Technical Memo TM-385, MIT Laboratory for Computer Science, February 1989.

[290] R. Sloan. *Computational Learning Theory: New Models and Algorithms.* Phd thesis, MIT Department of Electrical Engineering and Computer Science, 1989. Supervised by R. L. Rivest.

[291] M. Smith, N. Margolus, and T. Toffoli. Cellular automaton simulations of oil-and-water mixtures. To appear.

[292] M. A. Smith, H. Hrgovčić, N. Margolus, and T. Toffoli. A conformally invariant cellular automaton. To appear.

[293] W. Smith. PhD thesis, Department of Applied Mathematics, Princeton University, 1988.

[294] K. Steele and R. Soley. *Virtual Memory on a Dataflow Computer.* Computation Structures Group Memo 289, MIT Laboratory for Computer Science, July 1988.

[295] C. Stein. Efficient algorithms for bipartite network flow. Princeton University, 1987. Unpublished manuscript.

[296] W. W. Tait. A realizability interpretation of the theory of species. In R. Parikh, editor, *Logic Colloquium, '73*, pages 22–37, Volume 453 of Lecture Notes in Mathematics, Springer-Verlag, 1975.

[297] G. Taylor, et al. Evaluation of the spur Lisp architecture. In *13 th International Symposium on Computer Architecture*, pages 444–452, Tokyo, Japan, June 1986.

[298] T. Toffoli. *Cellular Automata Machines as Physics Emulators*, pages 154–160. World Scientific, 1988.

[299] T. Toffoli. Four topics in lattice gases: ergodicity; relativity; information flow; and rule compression for parallel lattice-gas machines. In *Proceedings of Discrete Kinetic Theory, Lattice Gas Dynamics and Foundations of Hydrodynamics*, Institute for Scientific Interchange, Turin, Italy, September 1988.

[300] T. Toffoli. Pattern recognition and tracking by texture-locked loops. Working paper. To appear in expanded form as a chapter of *Advanced Computer Architectures for Robotics and Machine Intelligence: Neural Networks and Neurocomputers.*

[301] K. Traub, S. Brobst, J. Hicks, G. Papadopoulos, A. Shaw, and J. Young. *Monsoon Software Interface Specifications.* Computation Structures Group Memo 296, MIT Laboratory for Computer Science, 1989. Also available as Motorola Cambridge Research Center Technical Report 1.

[302] M. R. Tuttle. A game-theoretic characterization of eventual common knowledge. October 1988. Unpublished manuscript.

[303] E. Upfal. An $O(\log N)$ deterministic packet routing scheme. In *Proceedings of the 21 st Annual ACM Symposium on Theory of Computing*, May 1989. To appear.

[304] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 18(11):1134–1142, 1984.

[305] I. Vuong-Adlerberg. *Cache for Multi-threaded Processors on a Split-Transaction Bus.* Technical Report MIT/LCS/TR-466, MIT Laboratory for Computer Science, November 1989.

[306] P. Wadler and S. Blott. How to make *ad hoc* polymorphism less *ad hoc*. In *Proceedings of the 16 th Annual ACM Symposium on Principles of Programming Languages*, 1989.

[307] R. Watson. Timer-based mechanisms in reliable transport protocol connection management. *Computer Networks*, 5:47–56, 1981.

[308] J. Welch, L. Lamport, and N. Lynch. A lattice-structured proof technique applied to a minimum spanning tree algorithm. In *Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing*, pages 28–43, Toronto, Canada, August 1988. Expanded version in Technical Memo MIT/LCS/TM-361, MIT Laboratory for Computer Science, June 1988.

[309] J. Welch and N. Lynch. Synthesis of efficient drinking philosophers algorithms. In progress.

[310] M. P. Wellman. *Formulation of Tradeoffs in Planning Under Uncertainty.* PhD thesis, MIT, July 1988.

[311] W. Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13:591–606.

[312] S. Wu. An efficient algorithm for finding two edge-disjoint paths in a graph. 1988. Submitted for publication.

[313] A. Xu. *A Fault-tolerant Network Kernel for Linda.* Technical Report MIT/LCS/TR-424, MIT Laboratory for Computer Science, August 1988.

[314] A. C. Yao. Theory and applications of trapdoor functions. *IEEE Transactions on Computers*, 80–91, 1982.

[315] A. Yehuda, B. Awerbuch, and E. Gafni. Applying static network protocols to dynamic networks. In *28 th Annual Symposium on Foundations of Computer Science*, IEEE, October 1987.

[316] P. Yew, N. Tzeng, and D. H. Lawrie. Distributed hot-spot addressing in large-scale multiprocessors. *IEEE Transactions on Computers*, C-36(14):388–395, April 1987.

[317] J. Young and Arvind. *Instruction Set Definition for an Explicit Token Store Machine.* Computation Structures Group Memo 293, MIT Laboratory for Computer Science, February 1989.

[318] V. Zue, J. Glass, M. Phillips, and S. Seneff. Acoustic segmentation and phonetic classification in the SUMMIT system. In *Proceedings from ICASSP*, pages 389–392, Glasgow, Scotland, 1989.

[319] V. Zue, J. Glass, M. Phillips, and S. Seneff. The MIT SUMMIT speech recognition system: a progress report. In *Proceedings from Speech and Natural Language Workshop*, pages 179–189, Philadelphia, PA, 1989.

OFFICIAL DISTRIBUTION LIST

DIRECTOR Information Processing Techniques Office Defense Advanced Research Projects Agency (DARPA) 1400 Wilson Boulevard Arlington, VA 22209	2 copies
OFFICE OF NAVAL RESEARCH 800 North Quincy Street Arlington, VA 22217 Attn: Dr. Gary Koop, Code 433	2 copies
DIRECTOR, CODE 2627 Naval Research Laboratory Washington, DC 20375	6 copies
DEFENSE TECHNICAL INFORMATION CENTER Cameron Station Alexandria, VA 22314	12 copies
NATIONAL SCIENCE FOUNDATION Office of Computing Activities 1800 G. Street, N.W. Washington, DC 20550 Attn: Program Director	2 copies
HEAD, CODE 38 Research Department Naval Weapons Center China Lake, CA 93555	1 copy